第三章
基本图形的扫描转换

计算机图形学

建模、变换、绘制和显示

显示

完成三维物体建模和绘制后，还要在显示器平面上将其呈现出来。首先借助投影变换绘制投影图，其次在图像显示器等光栅输出设备是图形用像素的集合来表示。（第3、4章）

本章主要内容：

- 3.1 直线的扫描转换
- 3.2 圆的扫描转换
- 3.3 椭圆的扫描转换
- 3.4 反走样技术
- 3.5 反走样Wu算法
- 3.6 本章小结
- 习题3

本章学习目标：

- 了解扫描转换的基本概念
- 熟练掌握直线的扫描转换算法
- 掌握圆的扫描转换算法
- 了解椭圆的扫描转换算法
- 熟练掌握Wu反走样算法

基本图形生成算法：光栅图形学

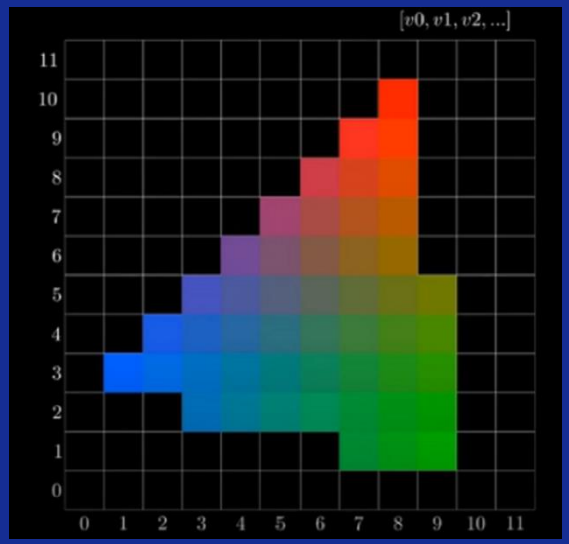
- 什么是光栅图形学？

  光栅显示器->图形光栅化、

  光栅化图形的处理；

  如何在光栅显示器上显示任何一种图形？

基本图形生成算法：光栅图形学

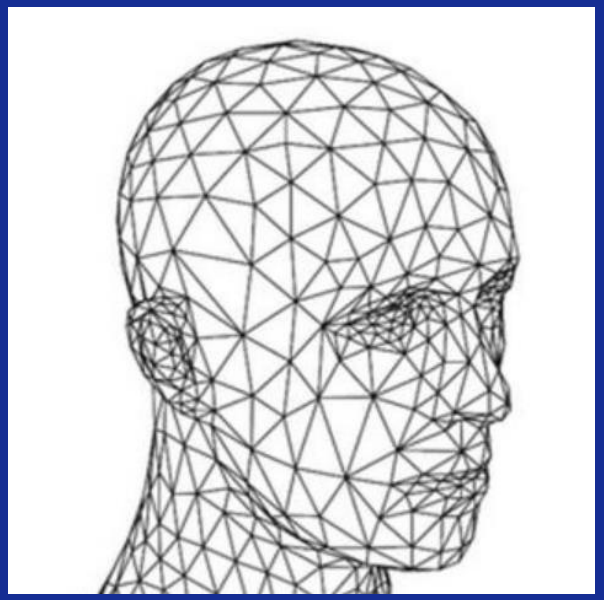


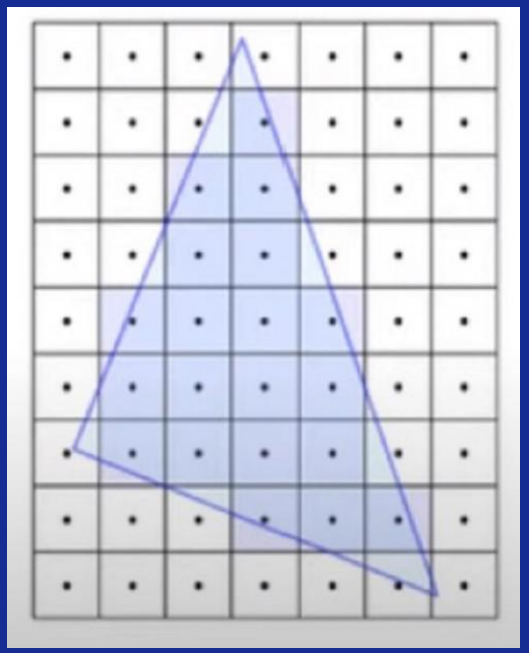
光栅图形显示器中电子束受偏转部件的控制，不断从左到右、从上到下将图像逐行逐点的扫描到显示屏上，通过控制电子束的强弱产生黑白、灰度或彩色的图像。

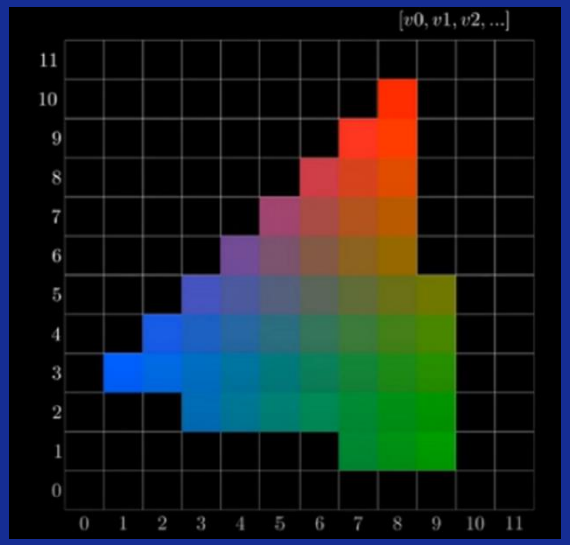
显示屏面扫描线为N行，每一行又可以分为M个点，则整个屏幕为M x N的点阵

每个由电子束选择的屏幕点称为个像素

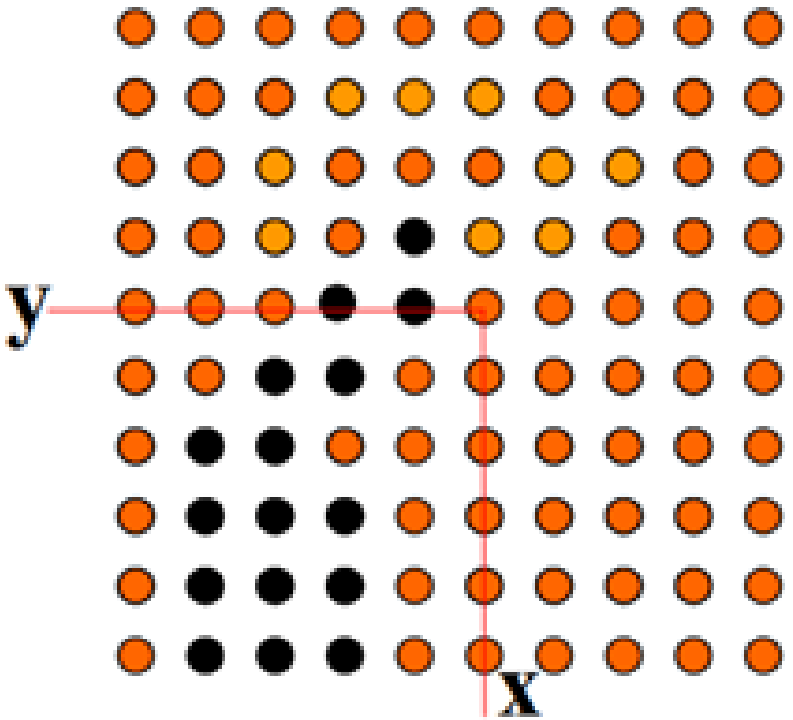
基本图形生成算法：光栅图形学



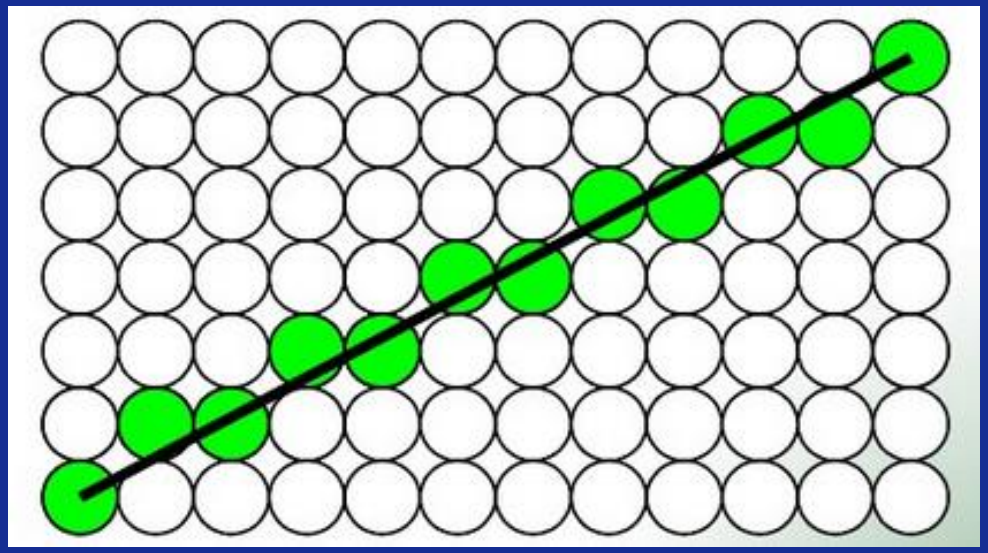




基本图形生成算法：光栅图形学



基本图形生成算法：光栅图形学



光栅显示器不能直接从单元阵列中的一个可编地址的像素画一条直线到另一个可编地址像素，只可能用尽可能靠近这条直线路径的像素点集来近似表示这条直线段。

基本图形生成算法：光栅图形学

■光栅化

基本图形光栅化就是在像素点阵中确定最佳逼近于理想图形的像素点集，并用制定颜色显示这些像素点集的过程。

■扫描转换

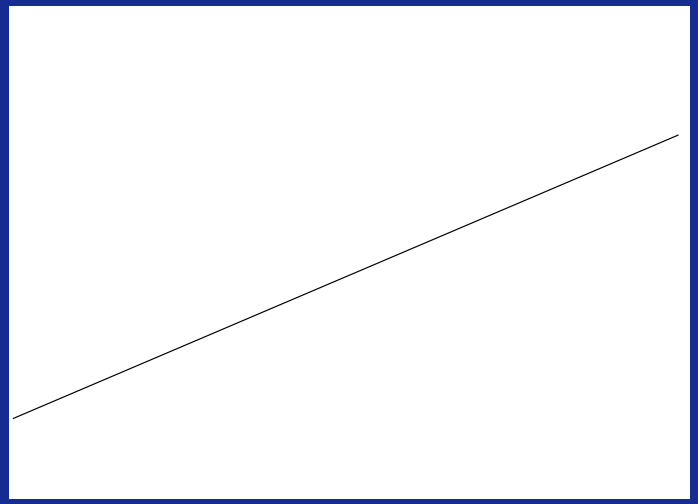
当光栅化与按扫描线顺序绘制图形的过程结合在一起时，也称为扫描转换

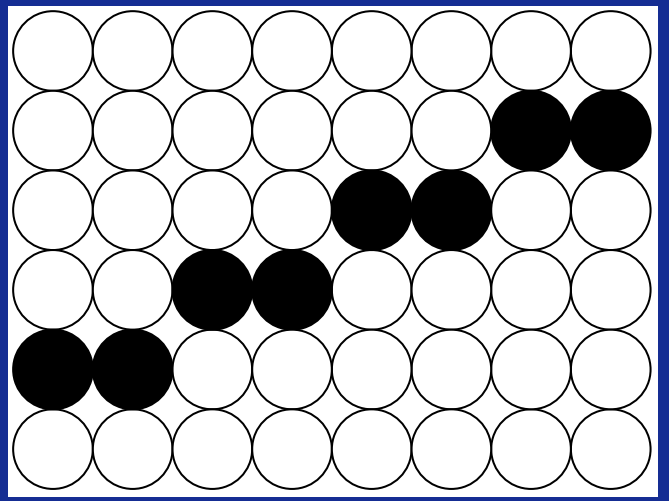
- **图形的扫描转换步骤**

  --确定相关的像素;

  --用图形的颜色或其他属性,对像素进行某种写操作;

- **直线y=kx+b表示**





基本图形生成算法：光栅图形学

- **一维图形**

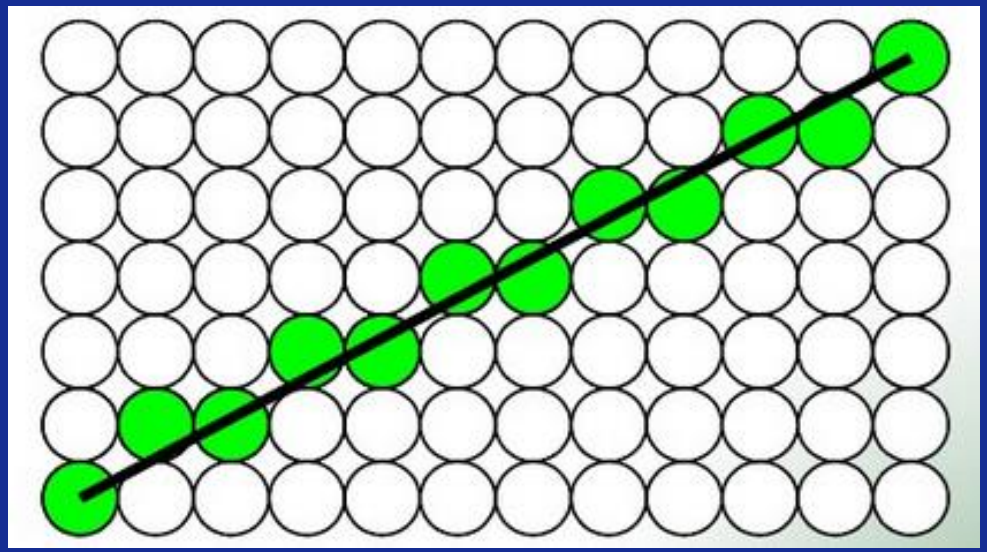
  用一个像素宽的直、曲线来显示图形。（直线段、圆弧）

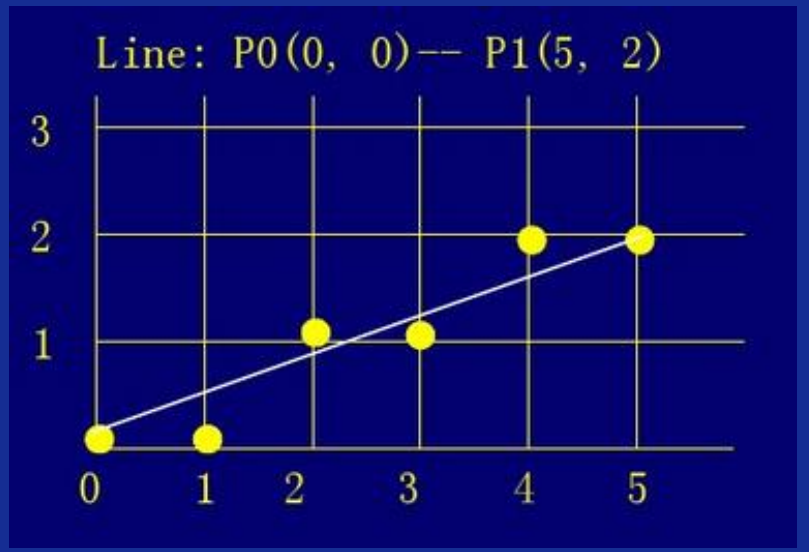
- **二维图形**

  必须确定区域对应的像素集，并用指定的属性或图形显示，即区域填充。（多边形的区域填充）

3.1 直线的扫描转换

- 在光栅显示器给定的有限个像素组成的矩阵中，确定一个最佳逼近于直线段的像素的集合，并用指定的属性写像素的过程。

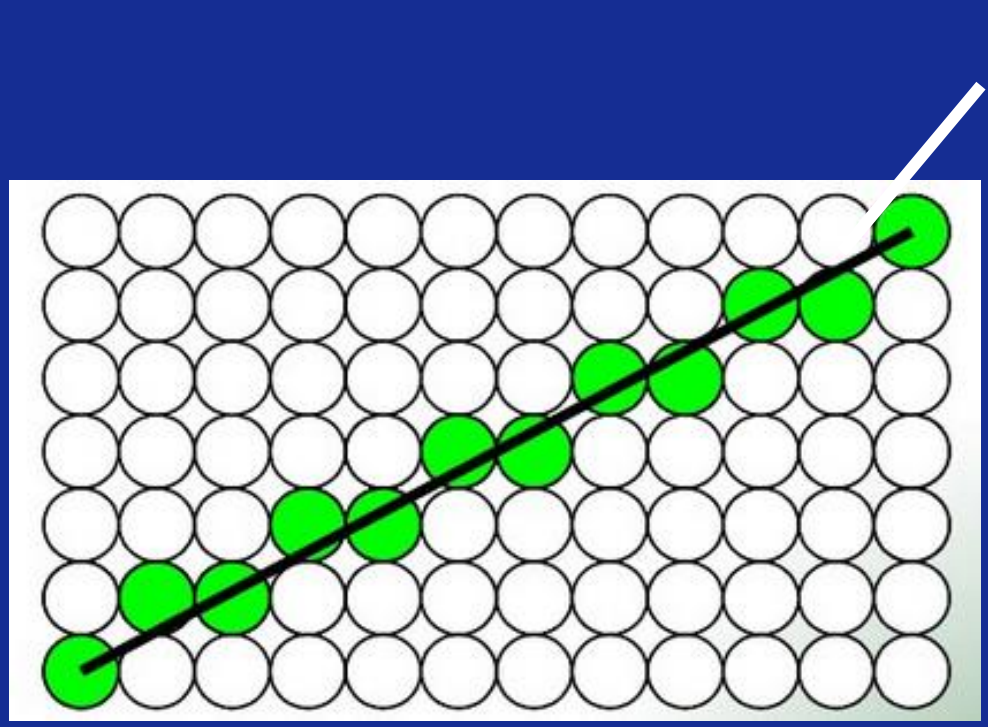




用一系列的像素点来逼近直线段，绿色的点是距离直线最近的像素

连续理想的直线扫描转换的结果是离散的像素点的集合

3.1 直线的扫描转换



- 找出最佳逼近这条直线段的所有点的坐标值

$$\{(x_0, y_0), (x_1, y_1), \ldots (x_{N-1}, y_{N-1})\}$$

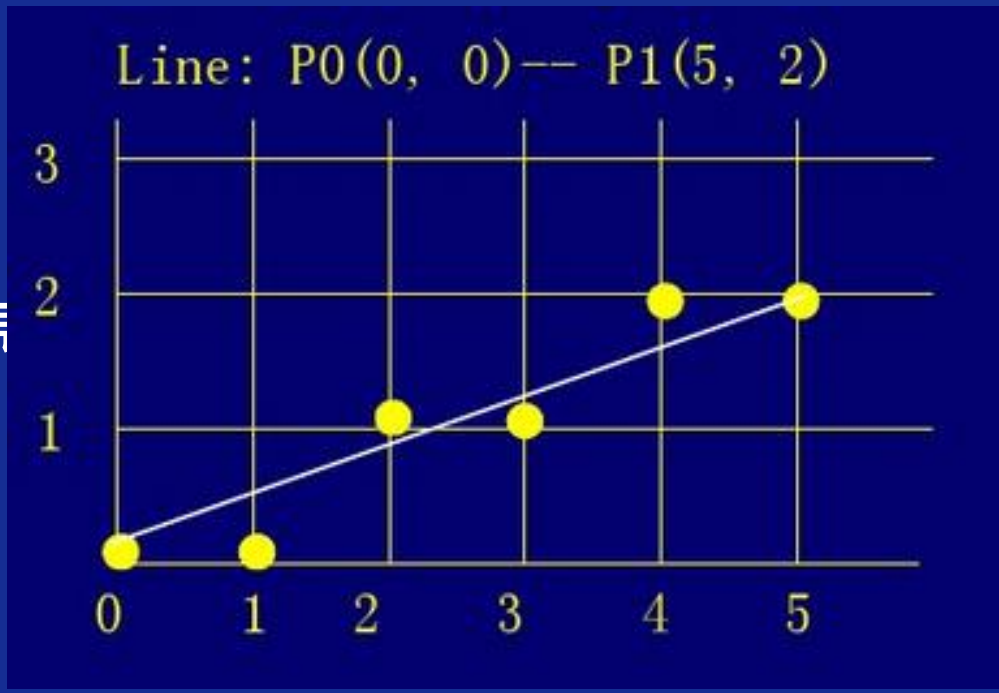
3.1直线的扫描转换

- 三个常用算法:

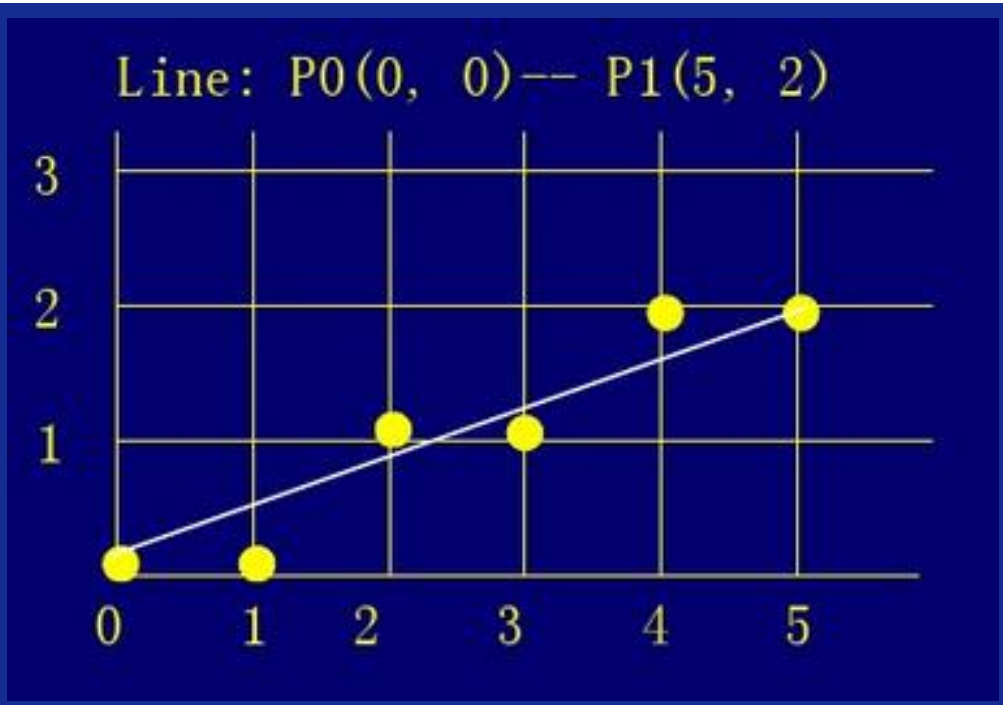
  数值微分法

  中点Bresenham算法

  Bresenham算法（课后扩展



直线段的表示



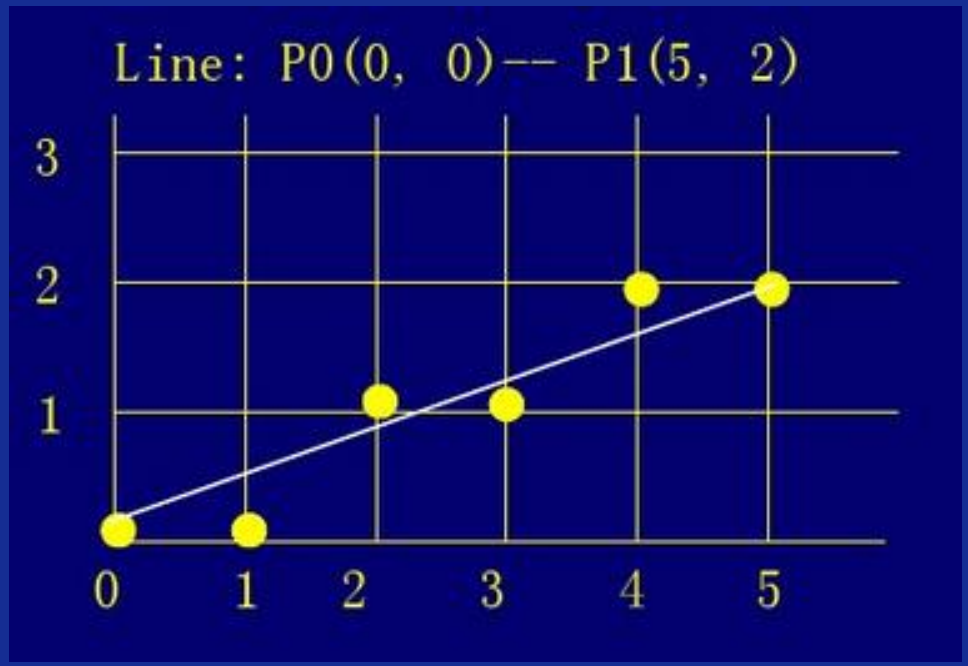
b是直线在y轴上的截距

已知过端点$P_0(x_0, y_0)$，$P_1(x_1, y_1)$的直线段L： $y = kx + b$

直线的**隐函数方程**为

$$F(x, y) = y - kx - b = 0$$

数值微分法



$$y_i = kx_i + b$$

$$x \leftarrow x_{i+1} = x_i + \Delta x$$

$$y \leftarrow y_{i+1} = kx_{i+1} + b$$

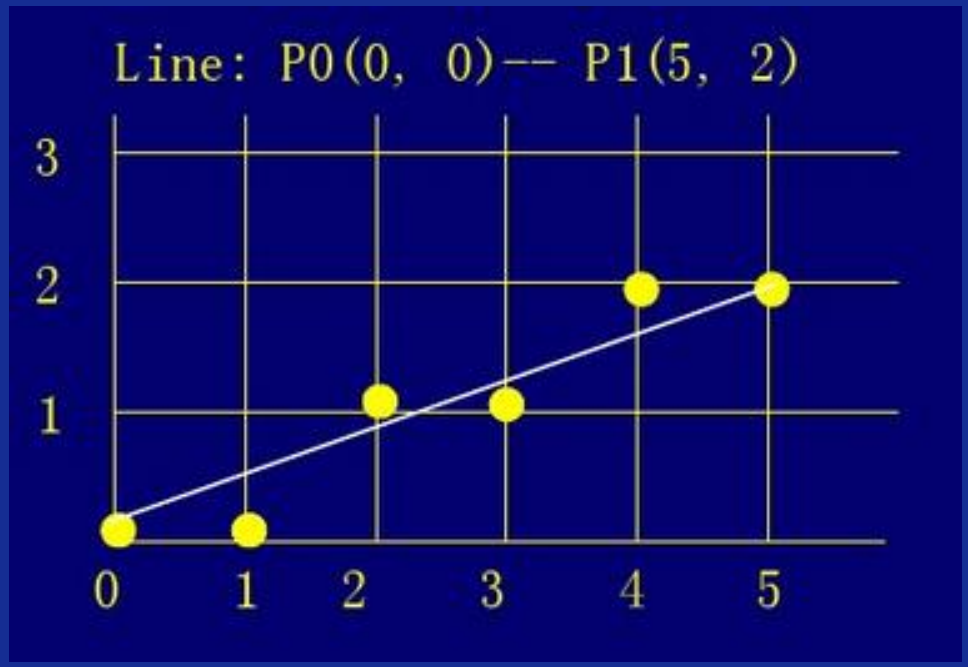
得到$(x, round(y))$

$$k = 0.4$$

这种方法直观，但效率太低，因为每一步需要一次浮点乘法和一次舍入运算。

数值微分法

要求绘制图形的速度要快，即尽量使用加减法，避免乘除、开方、三角等复杂运算

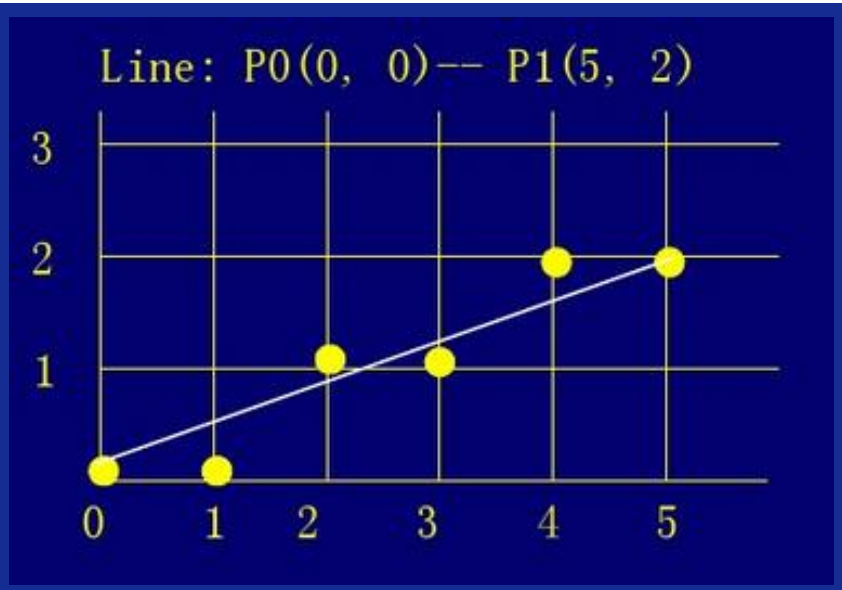


$k = 0.4$

$$x_{i+1} = x_i + \Delta x$$

$$y_{i+1} = kx_{i+1} + b = k(x_i + \Delta x) + b = kx_i + k\Delta x + b = y_i + k\Delta x$$

数值微分法



增量思想：每一步的x, y值是用前一步的值加上一个增量来获得的

当$\Delta x = 1$时

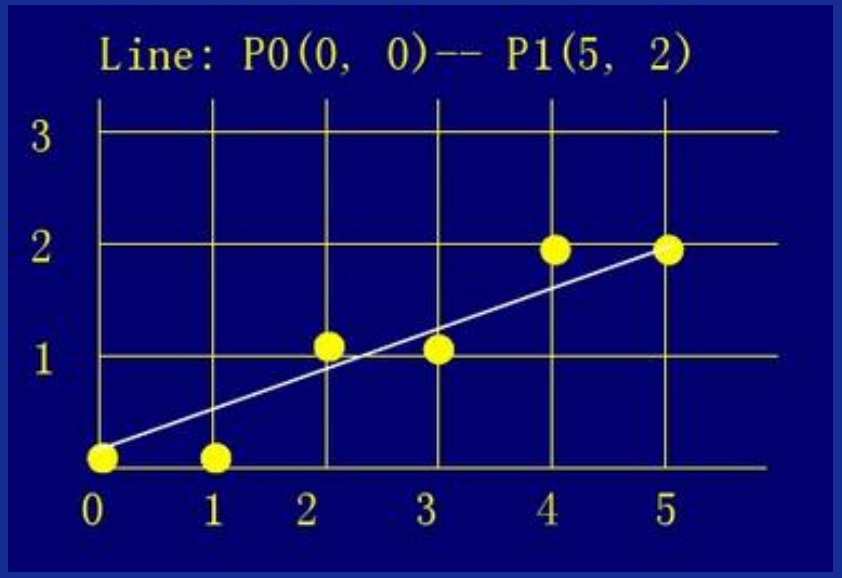
$$x_{i+1} = x_i + 1$$

$$y_{i+1} = kx_{i+1} + b = k(x_i + 1) + b = kx_i + k + b = y_i + k$$

一次浮点乘法和一次舍入运算变成一个加法运算

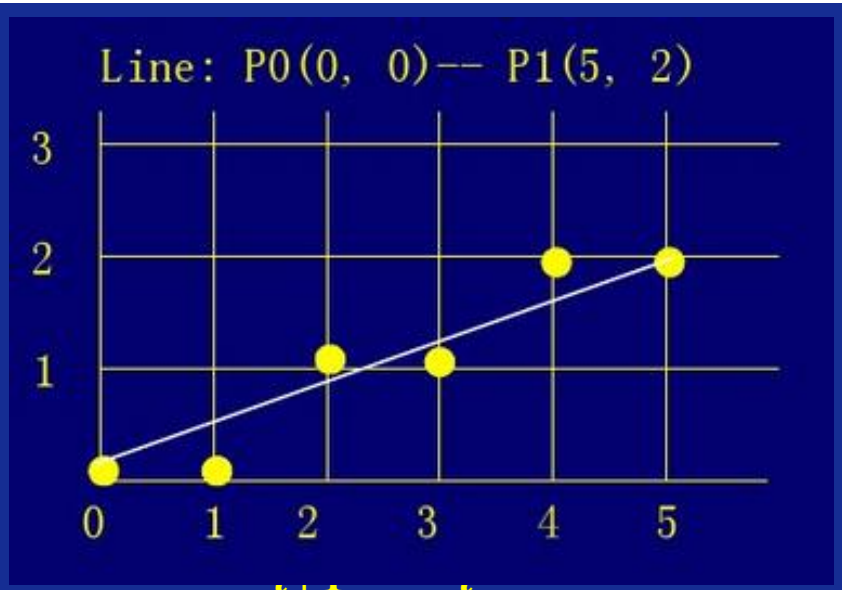
数值微分法

i	x_i	y_i	$\text{int}(y_i+0.5)$
0	0	0	0
1	1	0+0.4	0
2	2	0.4+0.4	1
3	3	0.8+0.4	1
4	4	1.2+0.4	2
5	5	1.6+0.4	2



$k = 0.4$

数值微分法



$k = 0.4$

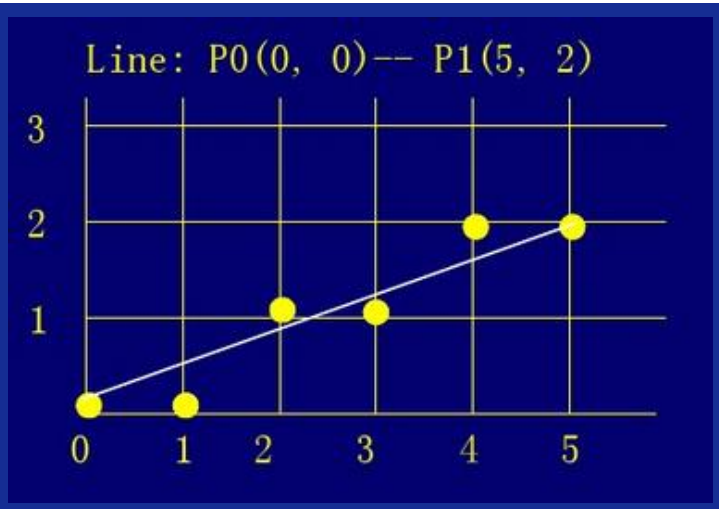
$$y_{i+1} = kx_{i+1} + b = k(x_i + 1) + b = kx_i + k + b = y_i + k$$

即：当x每递增1，y递增k(即直线斜率)；

注意上述分析的算法仅适用于|k| ≤1的情形。在这种情况下，x每增加1,y最多增加1。

当 |k| >1时，必须把x，y地位互换

3.1.1算法原理



已知过端点$P_0(x_0, y_0)$, $P_1(x_1, y_1)$的直线L： $y = kx + b$

直线斜率为

$$k = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y}{\Delta x}$$

$\Delta x = x_1 - x_0$是水平方向的位移，$\Delta y = y_1 - y_0$垂直方向的位移

假设直线斜率$0 \le k \le 1$,则有$\Delta x \ge \Delta y$,所以令x方向为主位移方向

如何判断M的位置

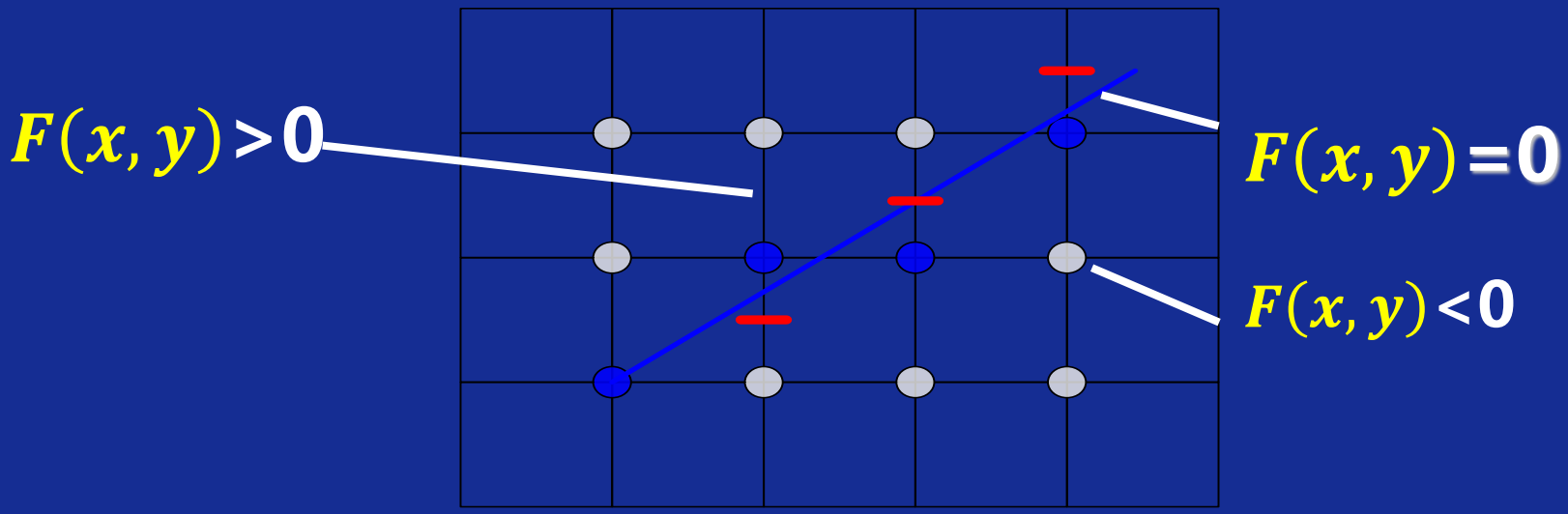
■对于理想直线段L（$p_0(x_0,y_0)$，$p_1(x_1,y_1)$

将平面分为三个区域,于是有下述点与L的关系：

--对于直线上的点：$F(x,y)=0$;

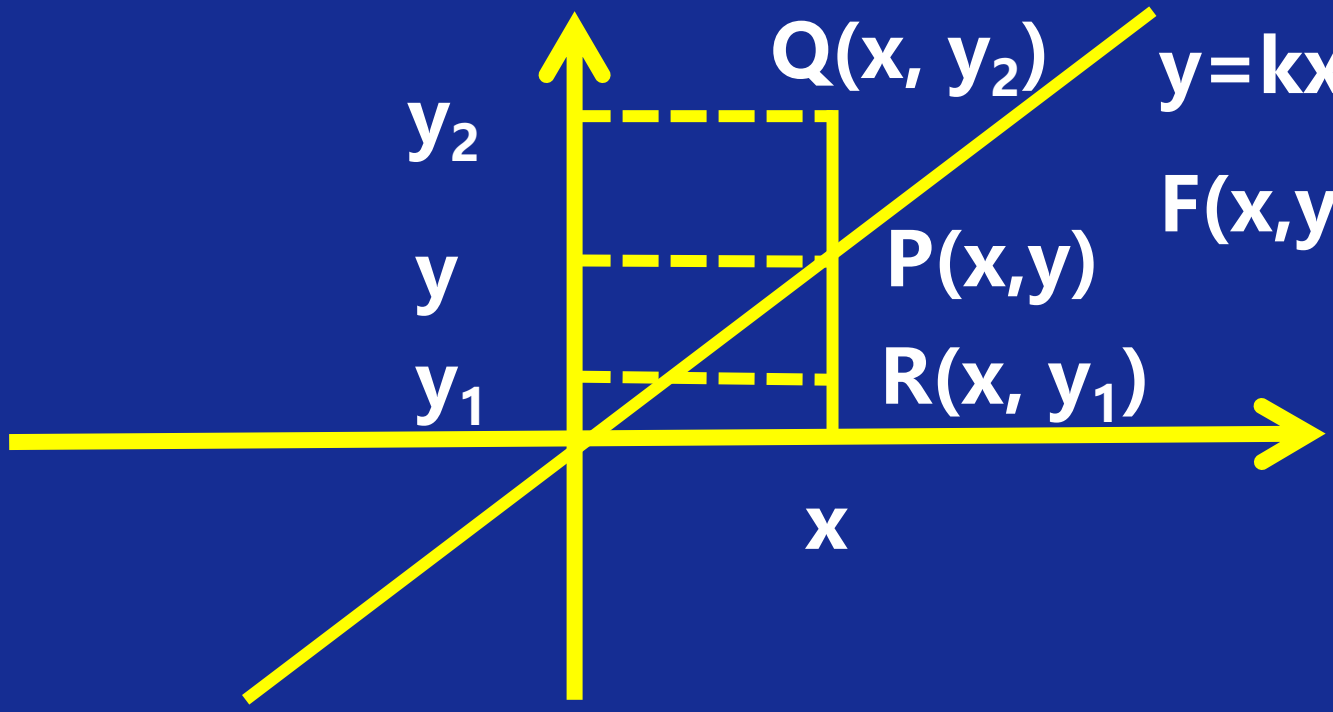
--对于直线上方的点：$F(x,y)>0$;

--对于直线下方的点：$F(x,y)<0$;



平面区域的划分

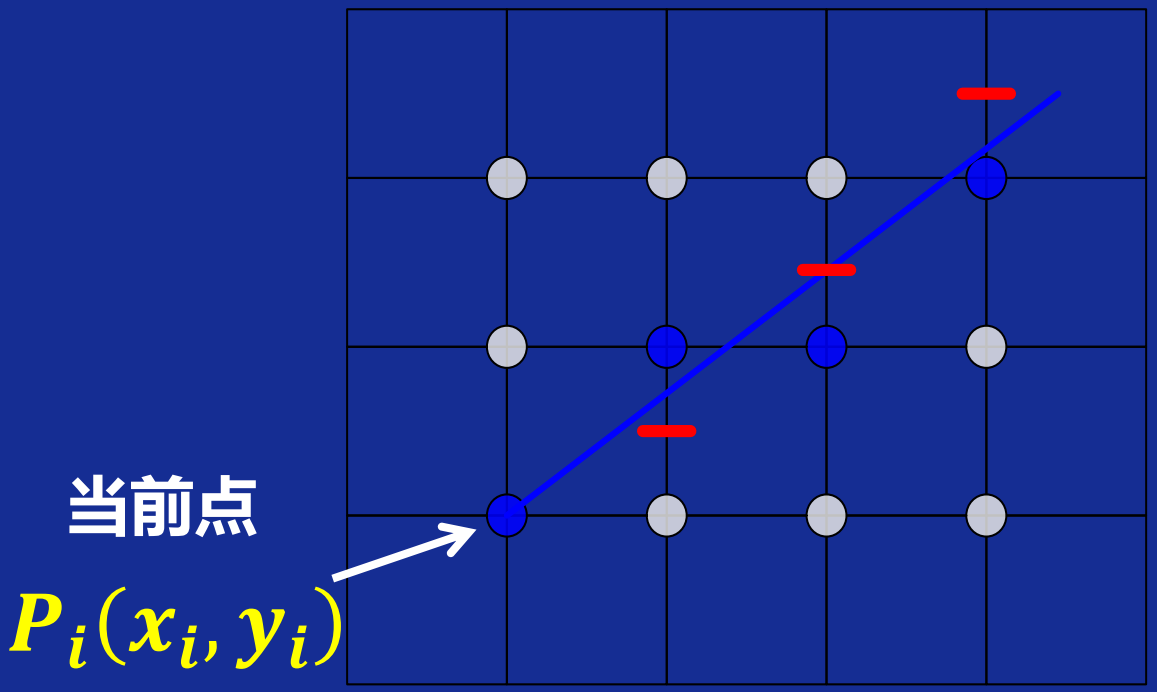
k>1



- $y_1 < y$,则有$y_1 < kx + b$可得 $y_1 - kx - b < 0$ 所以 $F(x, y_1) < 0$
- $y_2 > y$,则有$y_2 > kx + b$可得 $y_2 - kx - b > 0$ 所以 $F(x, y_2) > 0$

3.1.1直线中点Bresenham算法原理

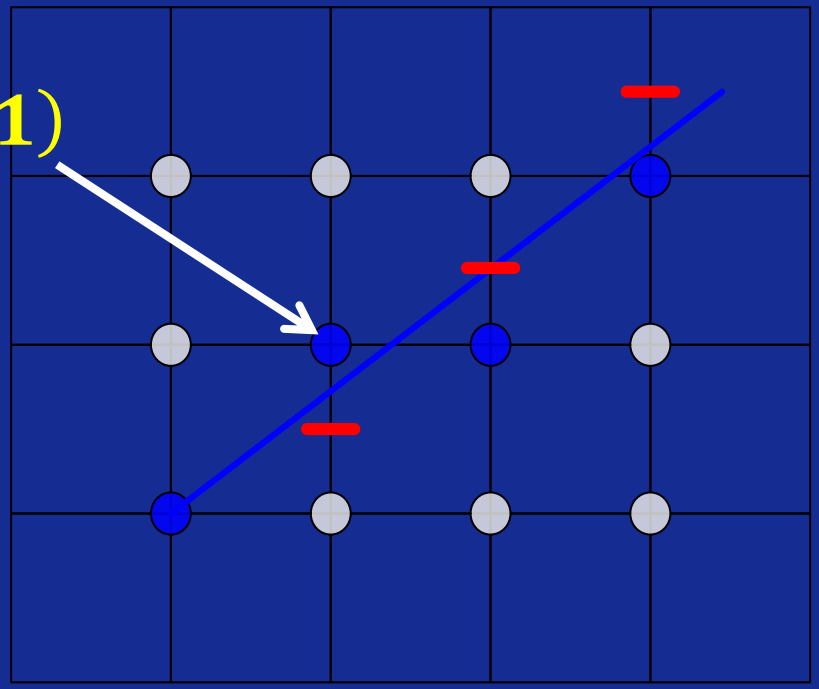
- 当前点的坐标、下一步点的坐标（步长为1）



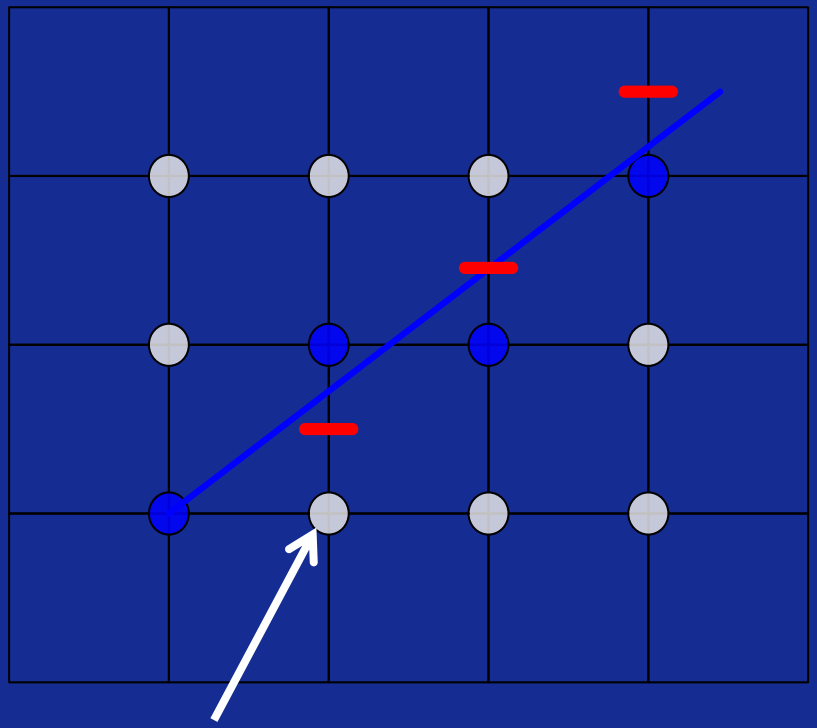
3.1.1直线中点Bresenham算法原理

右上点

$P_u(x_i+1, y_i+1)$

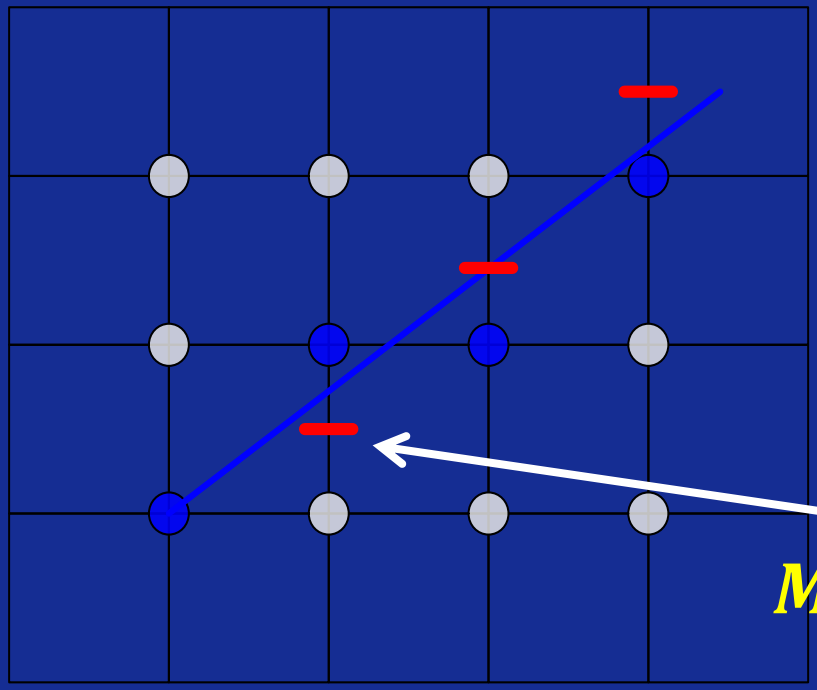


3.1.1直线中点Bresenham算法原理



右边点 $P_d(x_i+1, y_i)$

3.1.1直线中点Bresenham算法原理

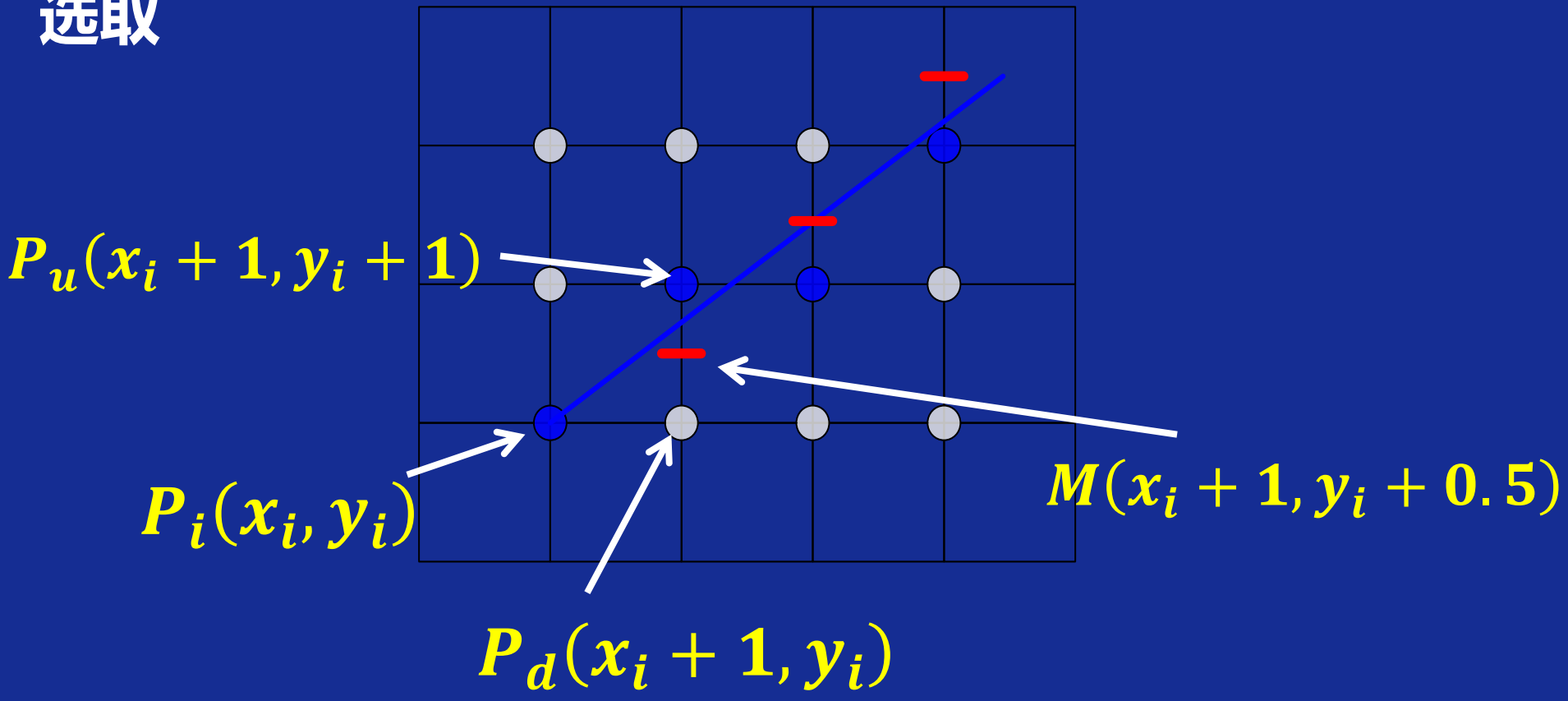


中间点

$M(x_i + 1, y_i + 0.5)$

3.1.1直线中点Bresenham算法原理

- 假定直线的当前点为 $P_i(x_i, y_i)$沿主位移x方向上走一步，下一点只能在 $P_u(x_i+1, y_i+1)$ 和 $P_d(x_i+1, y_i)$ 中选取



3.1.1直线中点Bresenham算法原理

当前像素点 $P_i(x_i, y_i)$下一个像素点P_u或者P_d

设$M = (x_i + 1, y_i + 0.5)$，为P_u与P_d的中点。

F（x，y）=0为理想直线

--当M在理想直线的下方，意味着 P_u距离直线近

则P_u应为下一个像素点；

--当M在理想直线的上方，意味着 P_d距离直线近

则P_d应为下一个像素点；

如何判断中心点M的在直线的下方还是上方?

3.1.2构造中点误差项

- 基本思想：由中点误差项的符号决定下一个像素取右边点还是右上点

从当前像素点$P_i(x_i，y_i)$出发选取下一像素点时，需将P_u与P_d的中点$M=（x_i+1，y_i+0.5）$带入隐函数方程，构造中点误差项d_i

$$d=F(M)=F(x_i+1,y_i+0.5)$$
$$=y_i+0.5-k(x_i+1)-b$$

如何判断M与L的位置

- 通过判断构造中点误差项符号判断M在直线L的上方还是下方

$$d_i = F(M) = F(x_i + 1, y_i + 0.5)$$
$$= y_i + 0.5 - k(x_i + 1) - b$$

--当$d_i < 0$,M在L下方， P_u距离直线近

则取右上方P_u应为下一个像素点；

--当$d_i > 0$,M在L上方，意味着 P_d距离直线近

则取右方P_d应为下一个像素点；

--当$d_i = 0$,选P_d或P_u均可，约定取P_d为下一个像素。

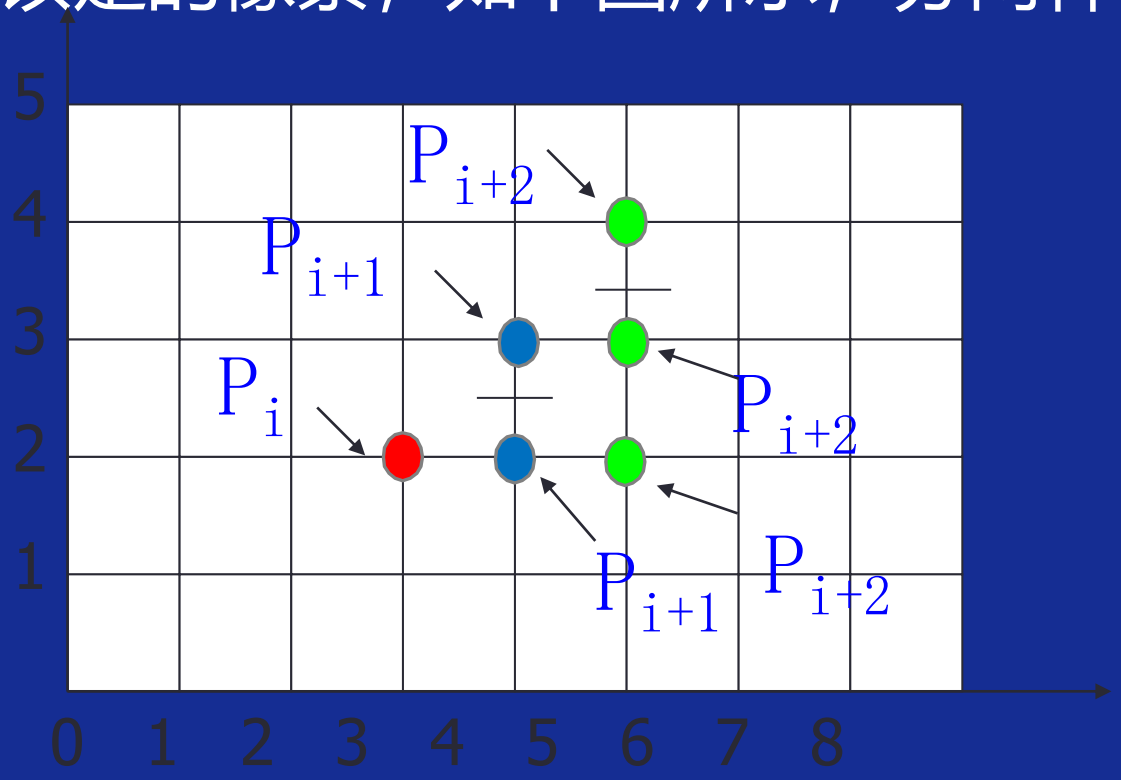
- 当前像素点 $P_i(x_i, y_i)$，那么下一像素点坐标 (x_{i+1}, y_{i+1})的选取具有以下性质：

  - 下一像素点的**横坐标**x_{i+1}一定是$x_i + 1$
  - 下一像素点的**纵坐标**y_{i+1}

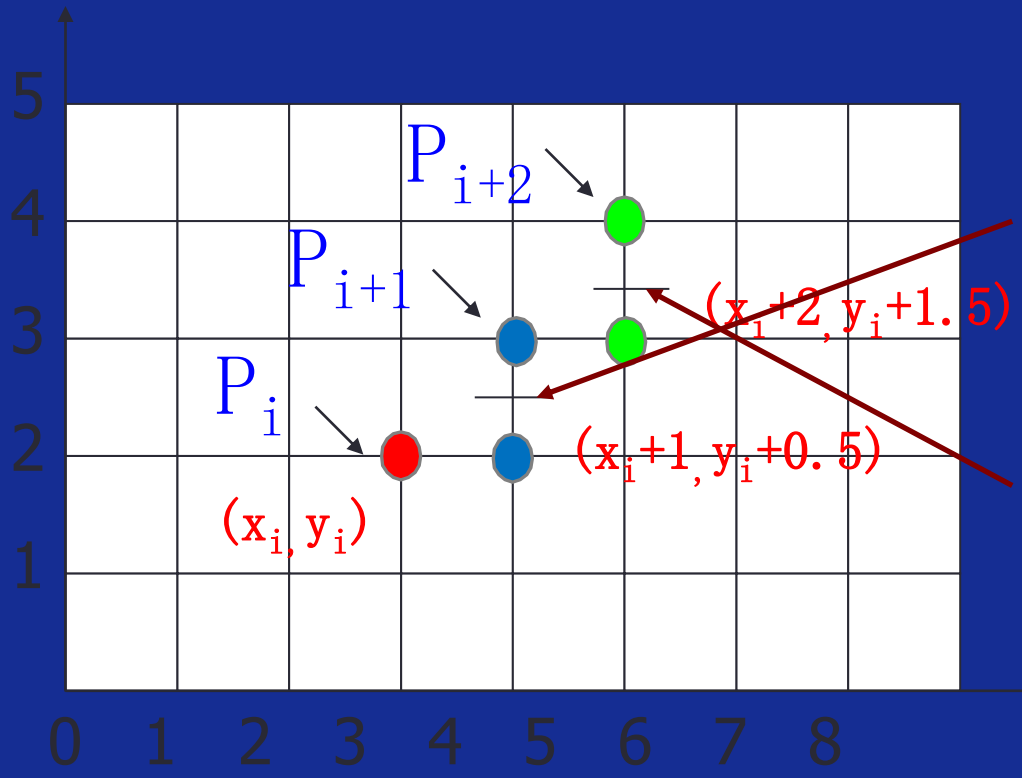
$$y_{i+1} = \begin{cases} y_i + 1, & d_i < 0 \\ y_i, & d_i \geq 0 \end{cases}$$

3.1.3 递推公式

- 在主位移x方向上已走一步的情况下，现在考虑沿主位移方向再走一步，应该选择哪个中点代入中点误差项以决定下一步该走的像素，如下图所示，分两种情况讨论。

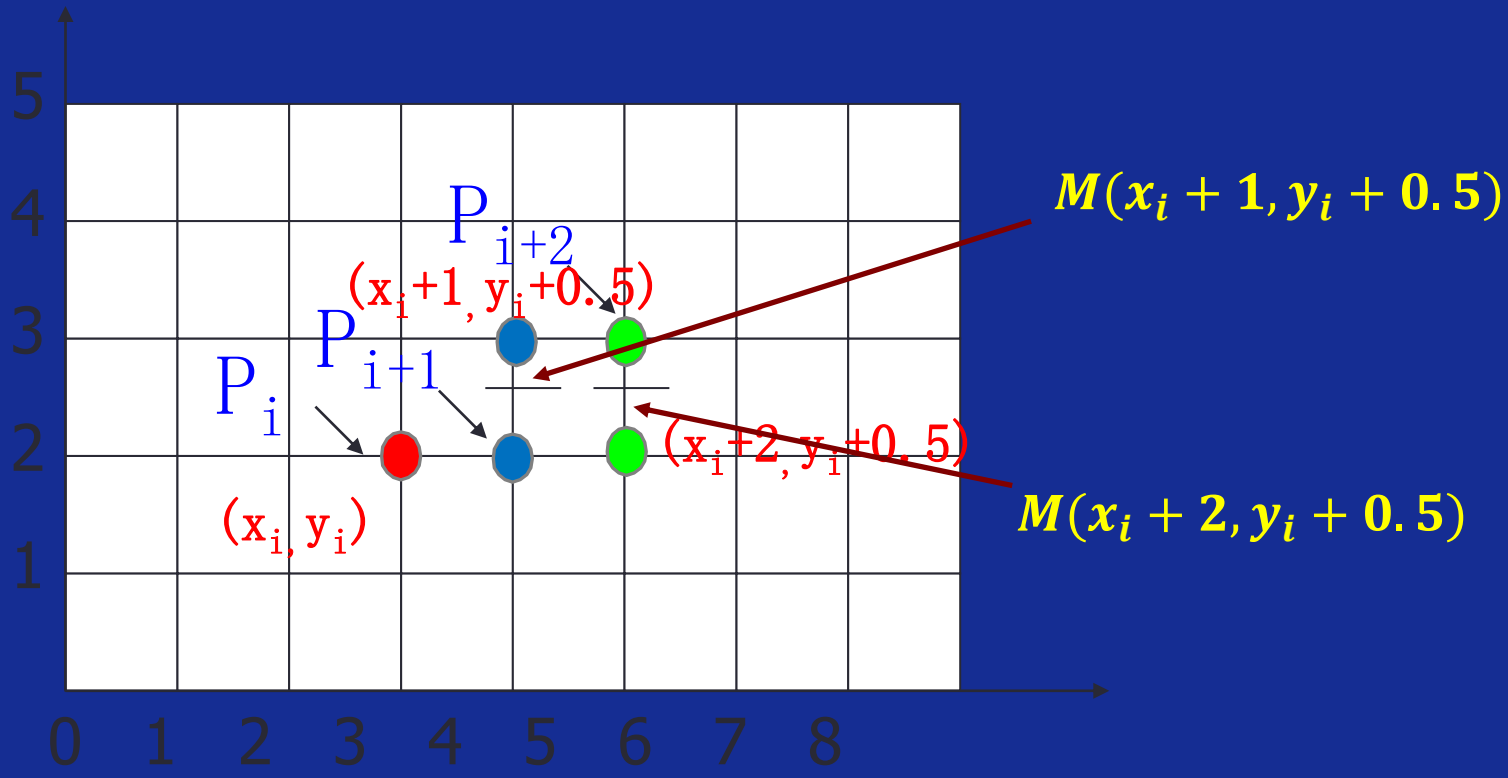


第一种情况



在此处键入公式。

第二种情况


5
4
3
2
1
0 1 2 3 4 5 6 7 8
P_{i+2}
$(x_i+1, y_i+0.5)$
P_{i+1}
P_i
(x_i, y_i)
$(x_i+2, y_i+0.5)$
$M(x_i+1, y_i+0.5)$
$M(x_i+2, y_i+0.5)$

3.1.3 递推公式

- 若$d_i < 0$ 情况，则取右上方像素$P_u(x_i + 1, y_i + 1)$,要判断再下一个像素位置，应计算

$$d_{i+1} = F(x_i + 2, y_i + 1.5)$$
$$= y_i + 0.5 - k(x_i + 1) - b + 1 - k;$$
$$= d_i + 1 - k$$

- 若$d_i \geq 0$ 情况，则取正右方像素$P_d(x_i + 1, y_i)$,要判断再下一个像素位置，应计算

$$d_{i+1} = F(x_i + 2, y_i + 0.5);$$
$$= d_i - k$$

3.1.3 中点误差项的初始值

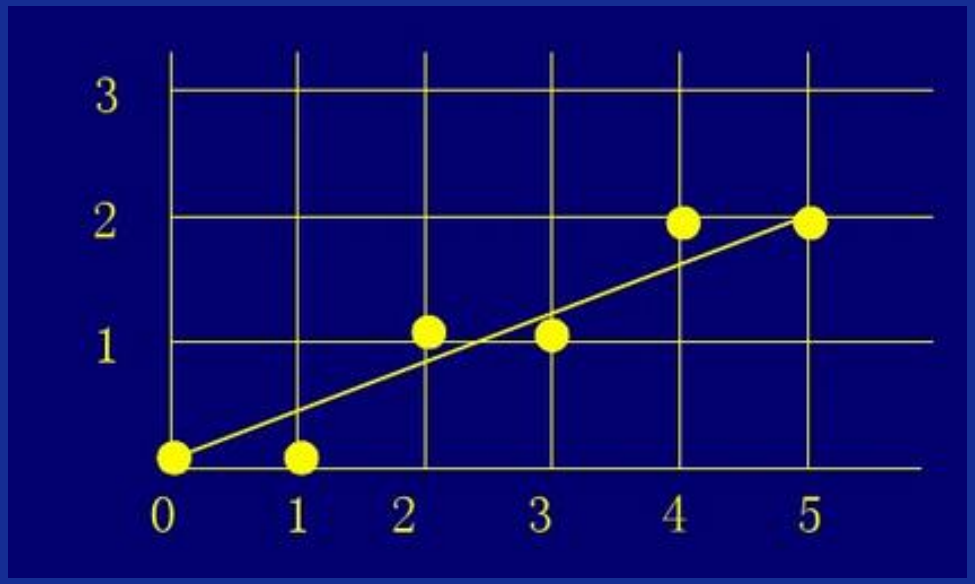
- 起点从$P_0(x_0, y_0)$开始,x为主位移方向。因此第一个中点是$M(x_0+1, y_0+0.5)$位移方向相应的d_i初始值为

$$d_0 = F(x_0+1, y_0+0.5)$$
$$= y_0 + 0.5 - k(x_0+1) - b'$$

因为$P_0(x_0, y_0)$ 在直线上，则$d_0 = 0.5 - k$

例子:中点画线法画直线段$P_0(0,0), P_1(5,2)$

- 找出近似该直线段的像素集合
-
-
-



例子:中点画线法画直线段$P_0(0,0), P_1(5,2)$

步骤：1、根据k 值确定主位移方向

- 2、写出中点误差项的初始值和地推公司

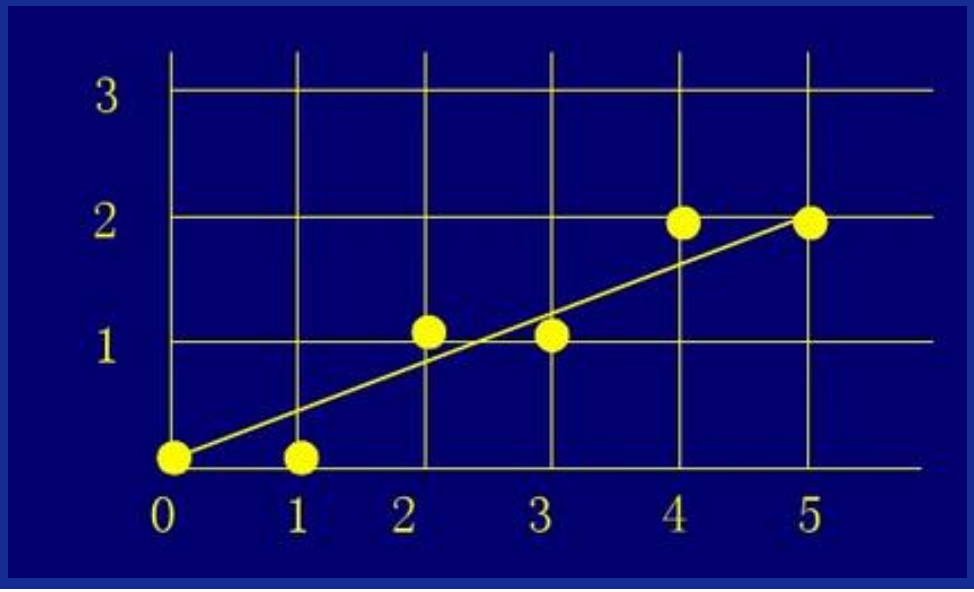
$$d_0 = 0.5 - k$$

$$d_i < 0 \qquad d_{i+1} = d_i + 1 - k$$

$$d_i \geq 0 \qquad d_{i+1} = d_i - k$$

- 3、找出所有像素点
- 完成下表

i	x_i	y_i	d_i



思考

■绘制任意斜率的直线

已知过端点$P_0(x_0, y_0)$, $P_1(x_1, y_1)$的直线L： $y = kx + b$

直线斜率为

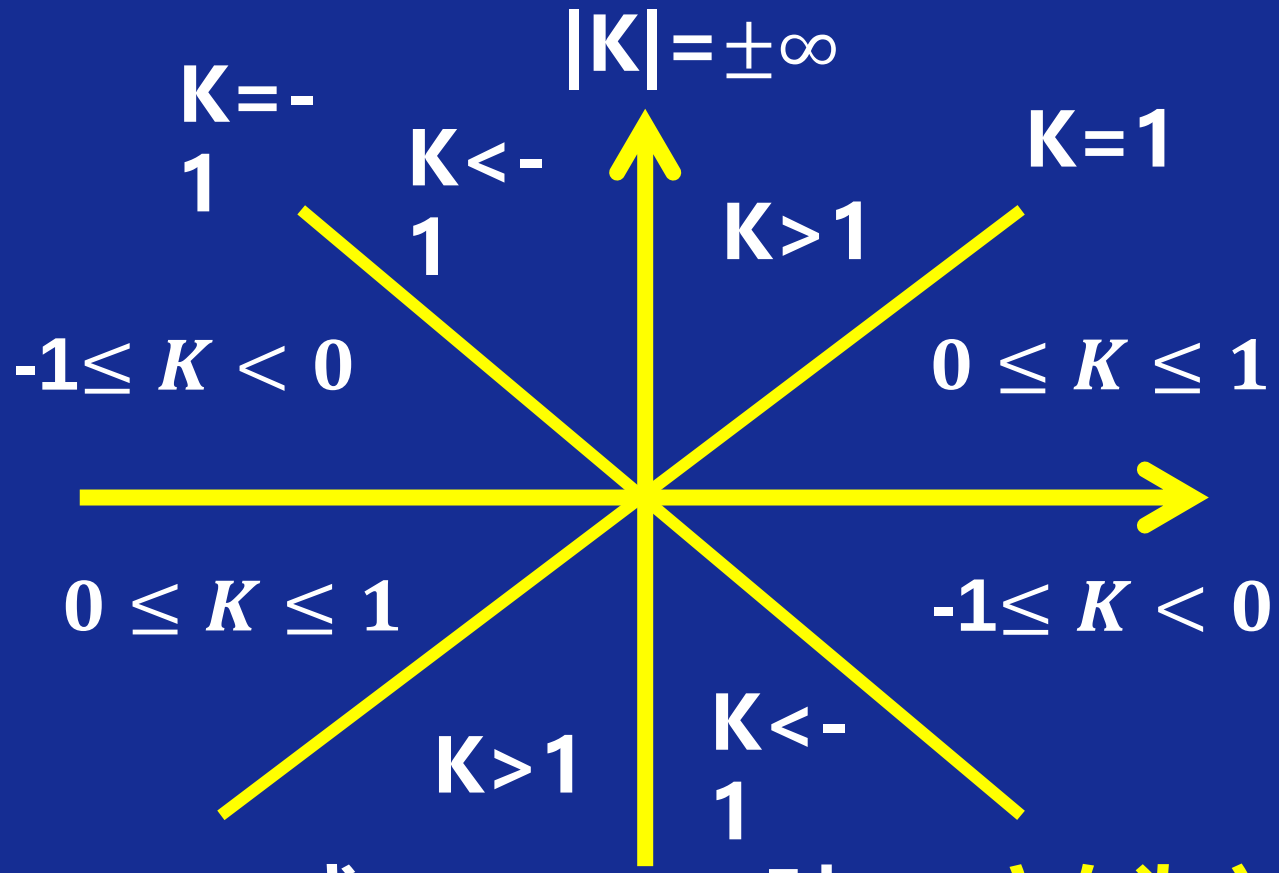
$$k = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y}{\Delta x}$$

$\Delta x = x_1 - x_0$是水平方向的位移，$\Delta y = y_1 - y_0$垂直方向的位移

假设直线斜率$k > 1$，则

- 哪个方向是主位移方向?
- 中点误差项的形式?
- 相应的中点误差项的递推公式？初始值的形式?

任意斜率



- 当 $0 \le K \le 1$ 或 $-1 \le K < 0$ 时，x方向为主位移方向
- 当 K>1 或 K<-1 时，y方向为主位移方向

扫描转换流程

- 第一步：**确定主位移方向**，因为由算法原理可知

  每次在主位移方向上走一步，另一个方向上走不走步（走多少步）取决于中点误差项的值。

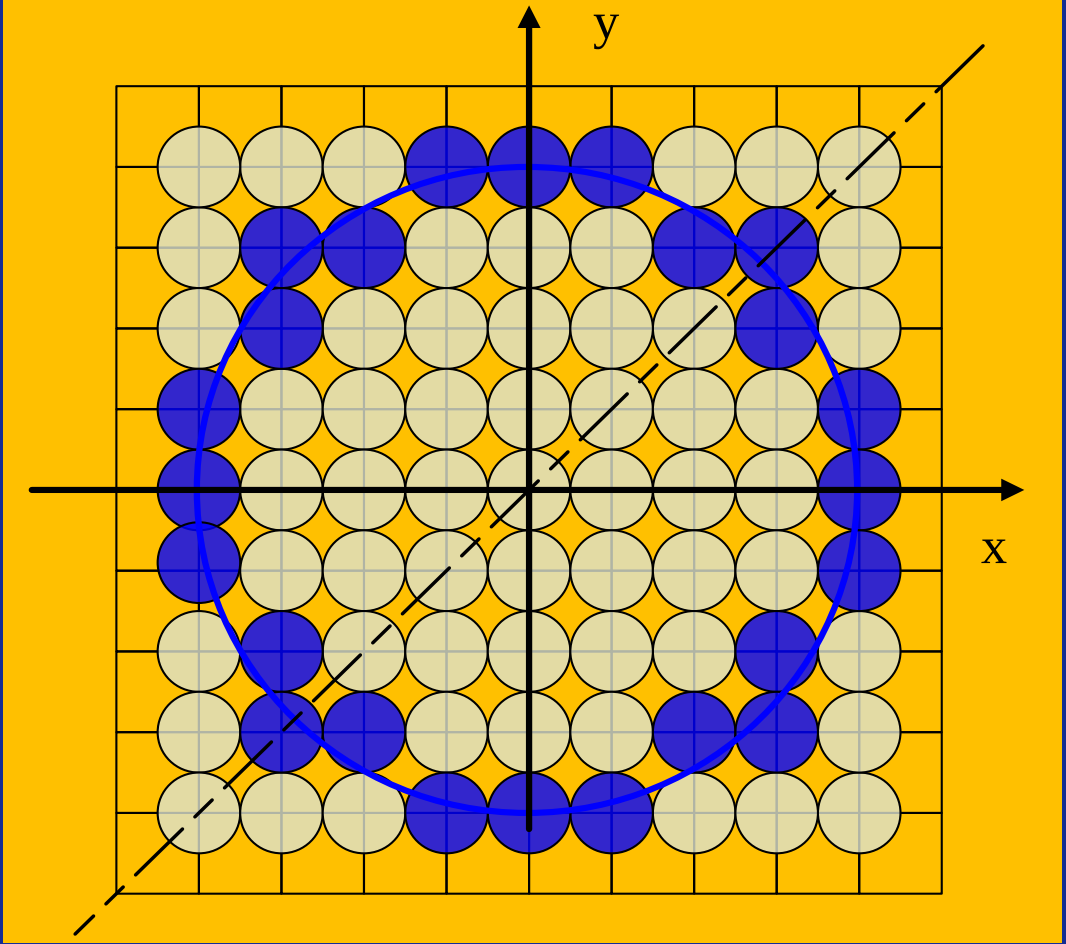
- 第二步：给定理想直线（圆、椭圆）的**隐函数方程**，将平面划分成三个区域：在理想直线（圆、椭圆）上的点、在理想直线（圆、椭圆）下方（内）的点、在理想直线（圆、椭圆）上方（外）的点、

- 第三步：给出选择下一像素的**中点误差项d**（将中点坐标带入隐函数），通过判断d的符号确定中点的与理想直线（圆、椭圆）的位置关系从而得出下一像素坐标

  即确定主位移方向前进一步，另一方向前不前进

3.2 圆的扫描转换

- 确定一个**最佳逼近于圆**的像素的集合



3.2 圆的扫描转换

- 圆的定义：到给定中心位置(x_c, y_c)的距离为r的点集。
- 圆的特征：八对称性。

圆心位于原点的圆有4条对称轴。若已知圆弧上一点(x, y)，可以得到其关于4条对称轴的其他7个点。

因此，只要扫描转换1/8圆弧，就可以利用八对称性求出整个圆弧的像素集。

3.2.1 算法原理

- 圆心在原点，半径为R的圆方程的隐函数表达式为

$$F(x,y) = x^2+y^2-R^2=0$$

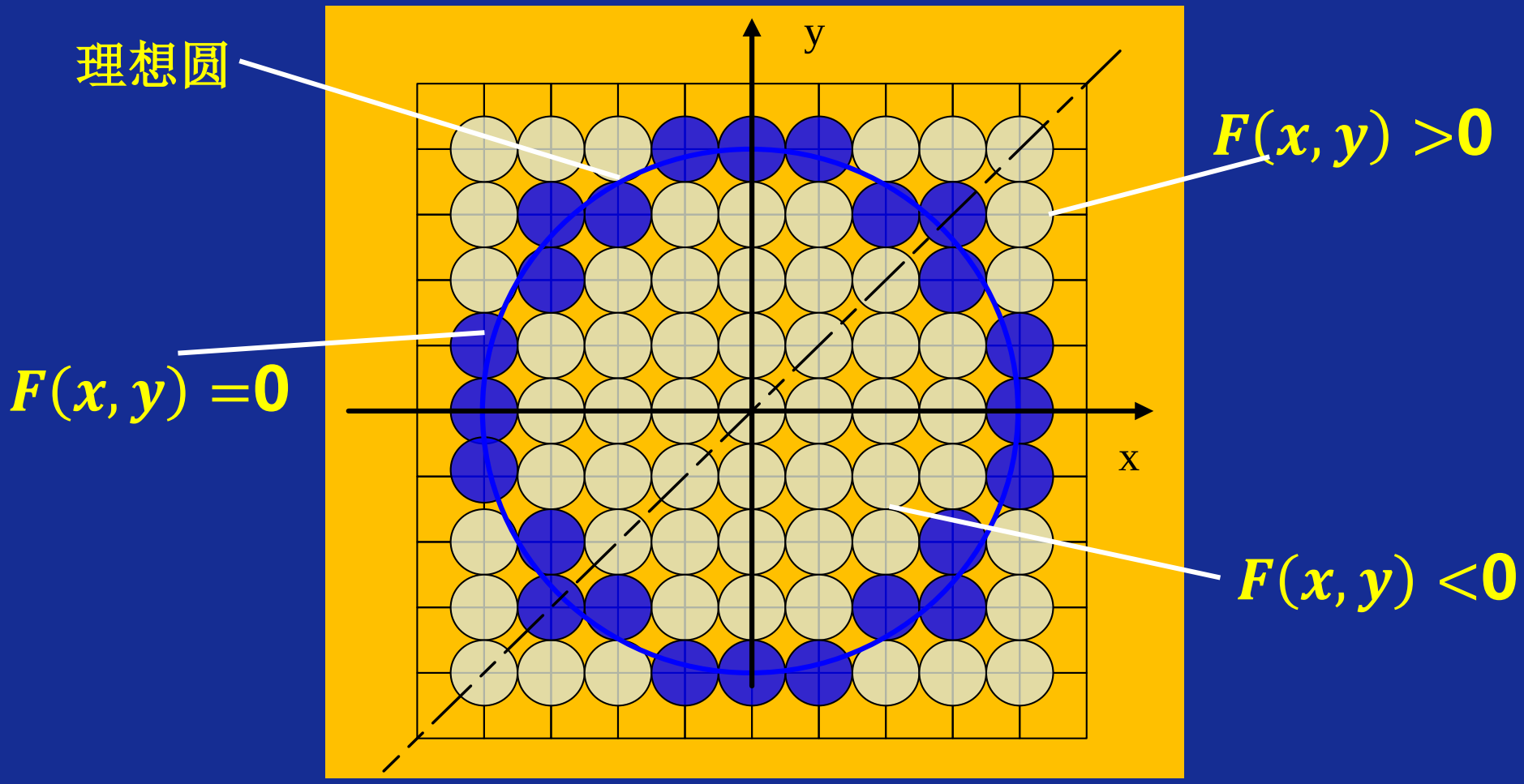
圆将平面划分成三个区域:

--圆上的点：$F(x,y)=0$;

--圆外的点：$F(x,y)>0$;

--圆内的点：$F(x,y)<0$;

3.2.1 算法原理



3.2.1 算法原理

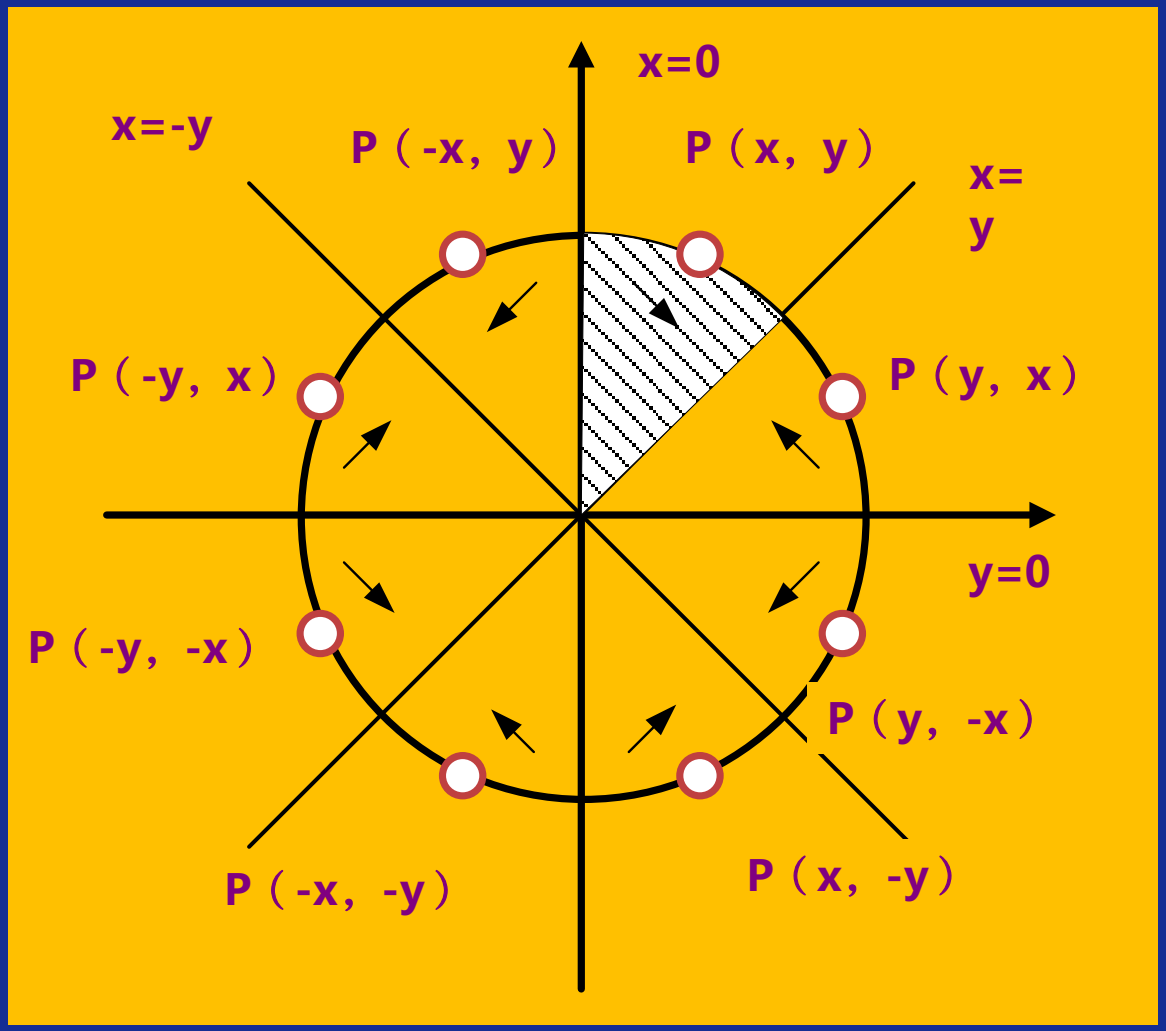
考虑到圆的对称性，可以利用四条对称轴**x = 0**，**y = 0**，**x = y**, **x = - y**把圆分成8等份。

只要绘制出第一象限内的1/8圆弧，根据对称性就可绘制出整圆，这称为八分法画圆算法。

假定第一象限内的任意点为P（x, y），可以顺时针确定另外7个点：

P（y，x），P（y，- x），P（x, - y），P（- x，- y），P（- y，- x），P（- y，x），P（- x，y）

3.2.1 算法原理



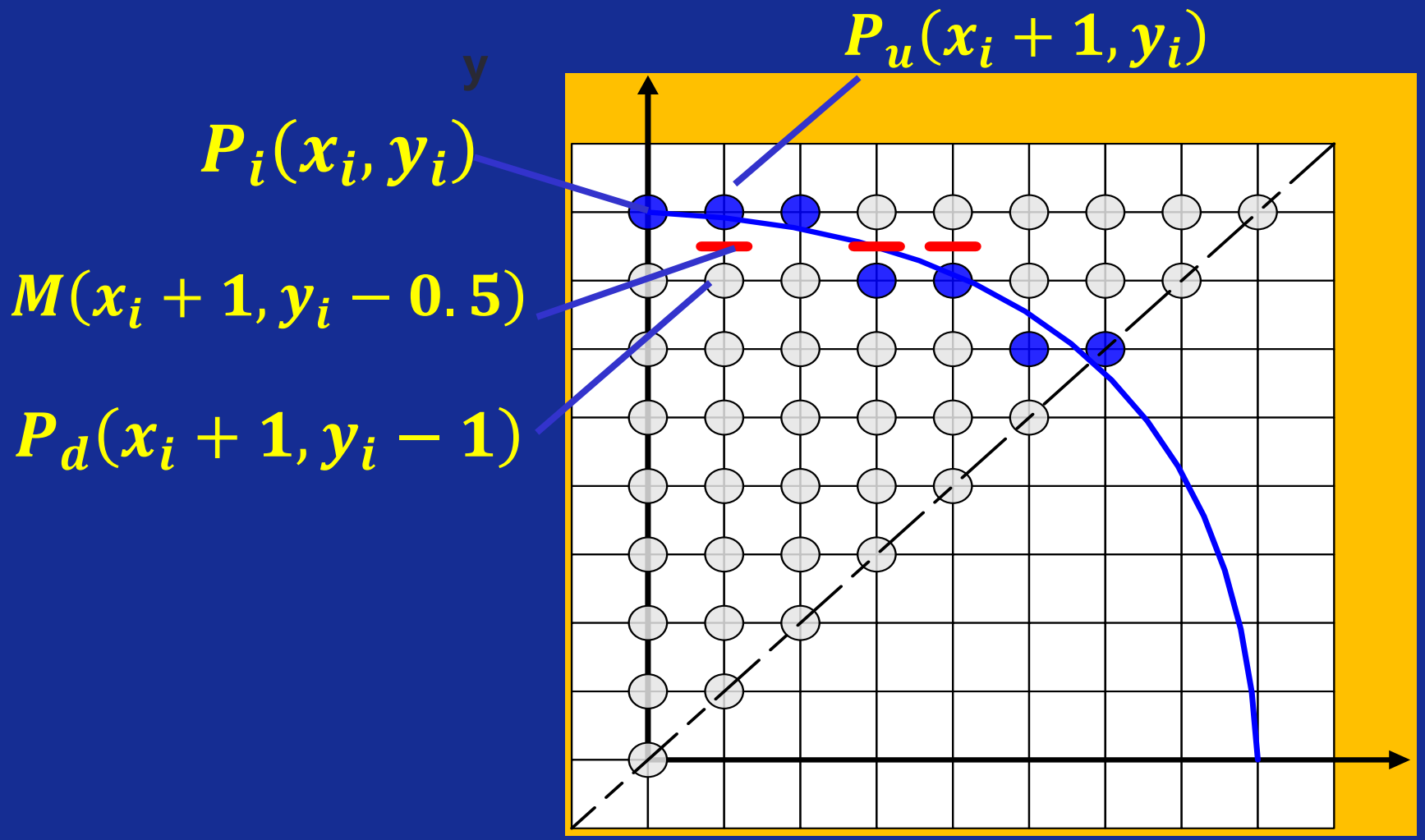
3.2.1 算法原理

- 本算法只考虑图中所示阴影部分的45°圆弧，即第一象限内 $x \in [0, \frac{R}{\sqrt{2}}]$ 的1/8圆弧
- 此时中点Bresenham算法要从 $(0, R)$ 到 $\left(\frac{R}{\sqrt{2}}, \frac{R}{\sqrt{2}}\right)$ 顺时针确定最佳逼近于该段圆弧的像素点集。

- 中点Bresenham算法的原理简化为：

x方向(主位移方向)上每次加1，y方向上减不减1取决于中点误差项的值。

3.2.1 算法原理



中点画圆法

- 假定圆当前点是 $P_i(x_i, y_i)$ ，

那么，下一点只能在 $P_u(x_i+1, y_i)$ 和$P_d(x_i+1, y_i-1)$中选取。P_u和P_d的中点为 $M(x_i+1, y_i-0.5)$

显然，若M点在理想圆弧的下方，则P_u点离圆弧近，则选取P_u；否则应选取P_d。

3.2.2构造中点误差项

- 基本思想：由中点误差项的符号决定下一个像素取右边点还是右上点

从当前像素点$P_i(x_i，y_i)$出发选取下一像素点时，需将P_u与P_d的中点$M=（x_i+1，y_i-0.5）$带入隐函数方程，构造中点误差项d_i

$$d=F(M)=F(x_i+1,y_i-0.5)$$
$$=(x_i+1)^2+(y_i-0.5)^2-R^2$$

如何判断M与圆弧的位置

- 通过判断构造中点误差项符号判断M在圆弧内还是圆弧外

$$d = F(M) = F(x_i + 1, y_i - 0.5)$$
$$= (x_i + 1)^2 + (y_i - 0.5)^2 - R^2$$

--当$d_i < 0$,M在圆弧内，意味着 P_u距离近

则y方向上不退步；

--当$d_i > 0$,M在圆弧外，意味着 P_d距离近

则y方向上退一步；

--当$d_i = 0$,选P_d或P_u均可，约定取P_d为下一个像素。

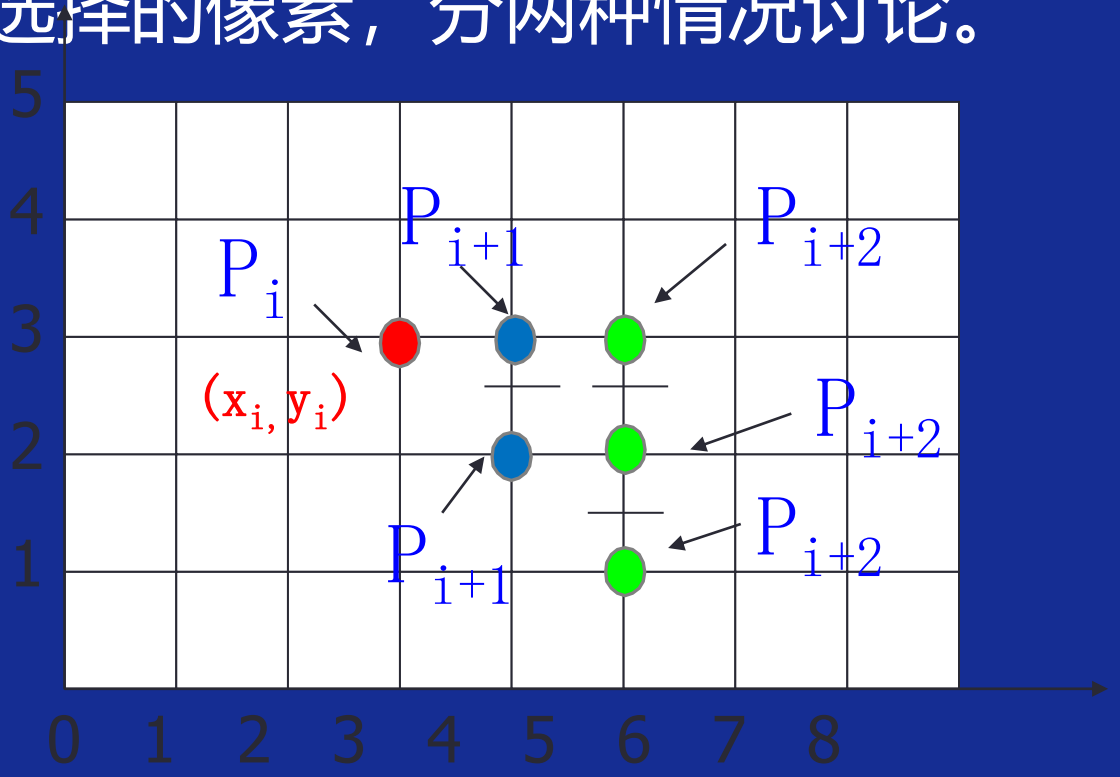
- 当前像素点 $P_i(x_i, y_i)$，那么下一像素点坐标 (x_{i+1}, y_{i+1})的选取具有以下性质：

  - 下一像素点的**横坐标**x_{i+1}一定是$x_i + 1$
  - 下一像素点的**纵坐标**y_{i+1}

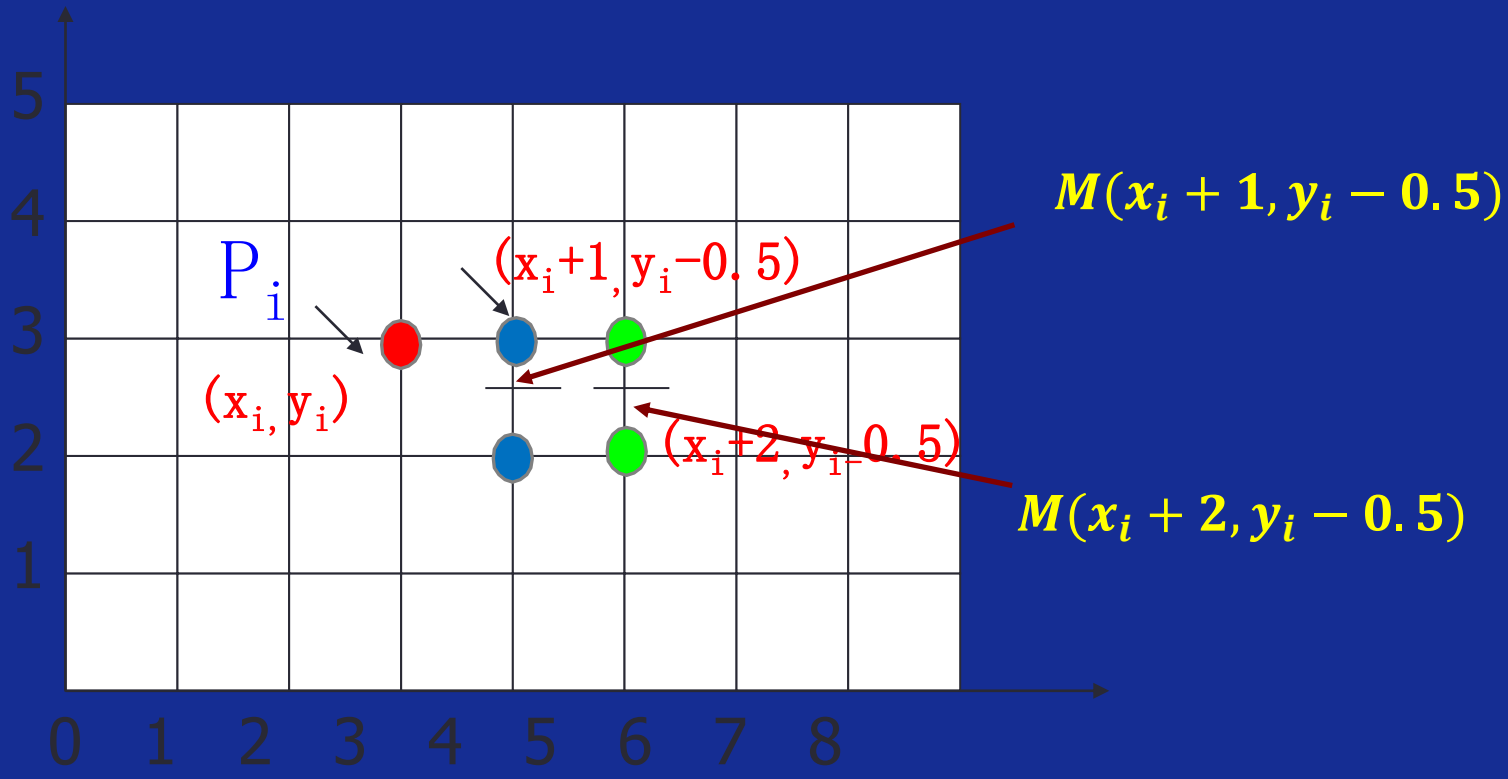
$$y_{i+1} = \begin{cases} y_i, & d_i < 0 \\ y_i - 1, & d_i \geq 0 \end{cases}$$

3.2.3 递推公式

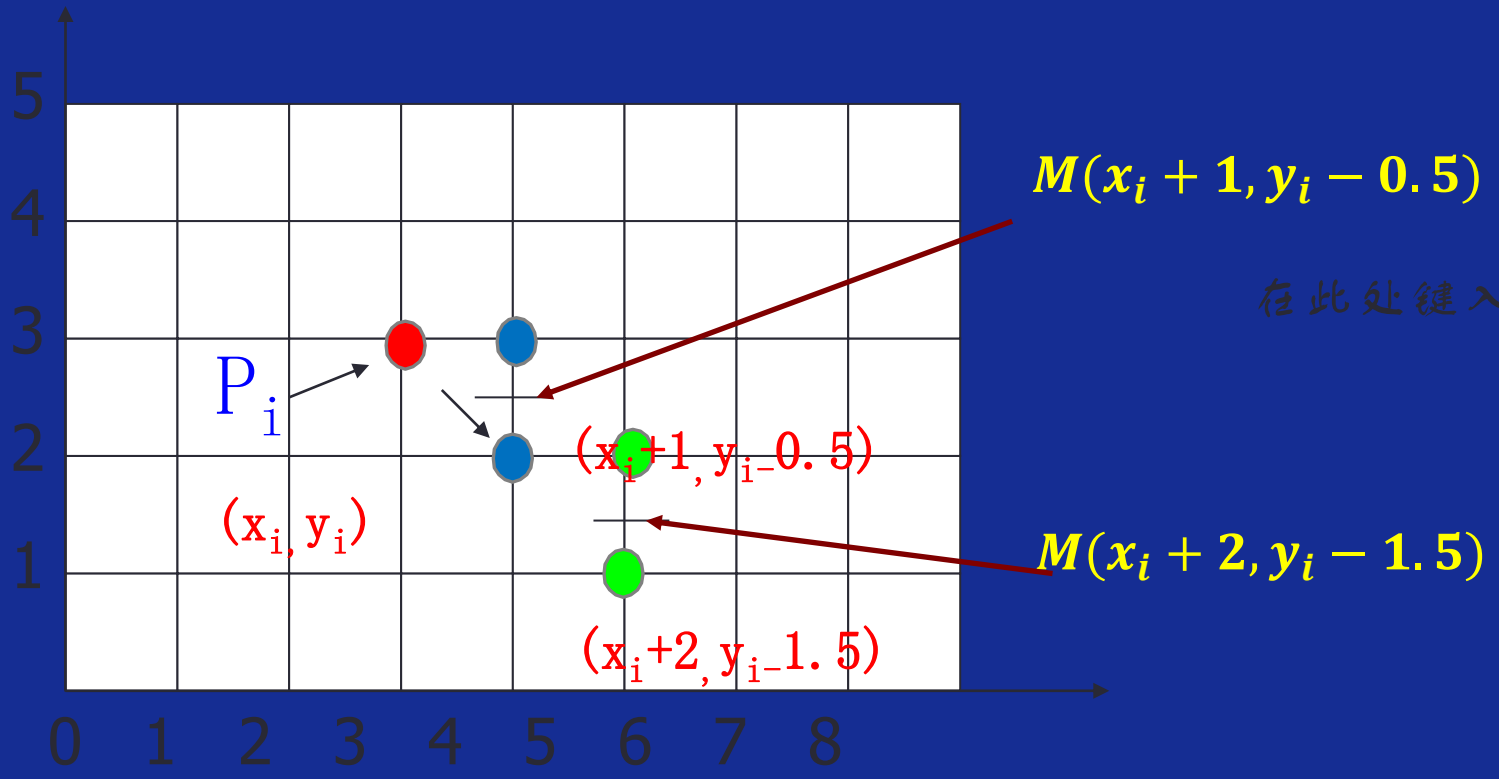
- 在主位移x方向上已走一步的情况下，现在考虑沿主位移方向再走一步，应该选择哪个中点代入中点误差项以决定下一步该选择的像素，分两种情况讨论。



第一种情况


5
4
3
2
1
0 1 2 3 4 5 6 7 8
P_i
(x_i, y_i)
$(x_i+1, y_i-0.5)$
$(x_i+2, y_i-0.5)$
$M(x_i+1, y_i-0.5)$
$M(x_i+2, y_i-0.5)$

第二种情况



在此处键入公式。

3.1.3 递推公式

- 若$d_i < 0$情况，下一步的中点坐标$M(x_i + 2, y_i - 0.5)$，要判断再下一个像素位置，应计算

$$d_{i+1} = F(x_i + 2, y_i - 0.5); \\ = d_i + 2x_i + 3;$$

- 若$d_i \geq 0$情况，下一步的中点坐标$M(x_i + 2, y_i - 1.5)$，要判断再下一个像素位置，应计算

$$d_{i+1} = F(x_i + 2, y_i - 1.5); \\ = d_i + 2(x_i - y_i) + 5;$$

3.1.3 中点误差项的初始值

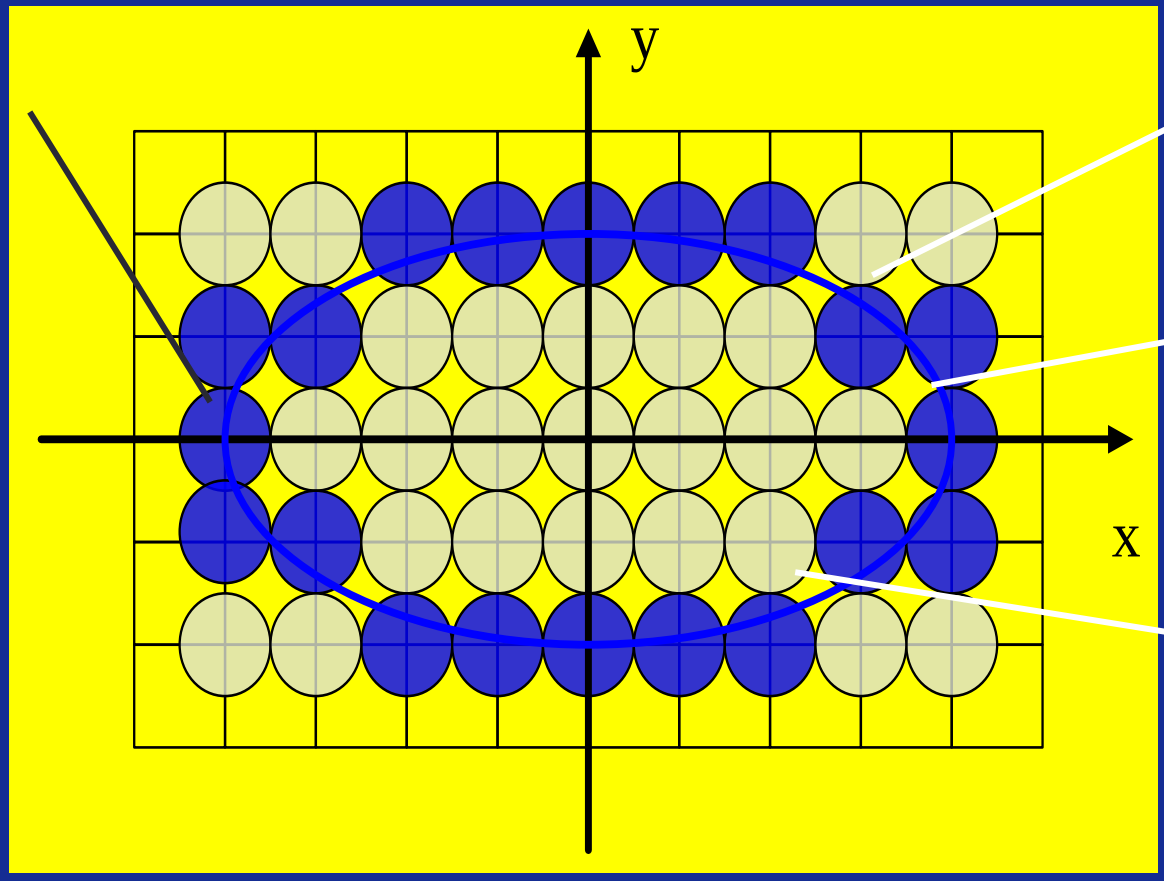
- 圆弧的起点P_0 的坐标为$(0, R)$,x为主位移方向。 因此第一个中点是$(1, R-0.5)$位移方向相应的d_i初始值为

$$d_0 = F((1, R-0.5)) = 1.25 - R$$

3.3 椭圆的扫描转换

- 确定一个**最佳逼近于椭圆**的像素的集合

理想椭圆



3.3.1 算法原理

- 圆心在原点，长半轴为a、短半轴为b的椭圆方程的隐函数表达式为：

$$F(x,y) = b^2x^2+a^2y^2-a^2b^2=0$$

椭圆将平面划分成三个区域:

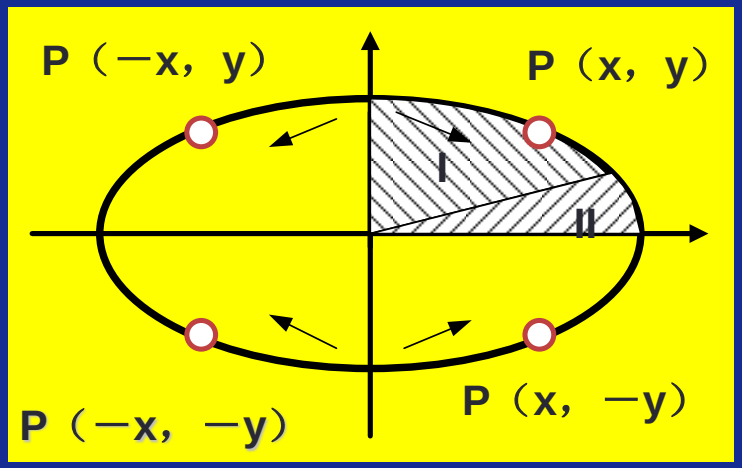
--椭圆上的点：$F(x,y)=0$;

--椭圆外的点：$F(x,y)>0$;

--椭圆内的点：$F(x,y)<0$;

3.3 椭圆的扫描转换

- 考虑到椭圆对称性，可以用对称轴x = 0，y = 0，把椭圆分成4等份。只要绘制出第一象限内1/4椭圆弧，如图的阴影部分Ⅰ和Ⅱ所示，根据对称性就可绘制出整个椭圆，这称为四分法绘制椭圆算法。已知第一象限内的点P（x, y），可以顺时针得到另外3个对称点：P（x, - y），P（ - x, - y），P（ - x, y）。



3.3 椭圆的扫描转换

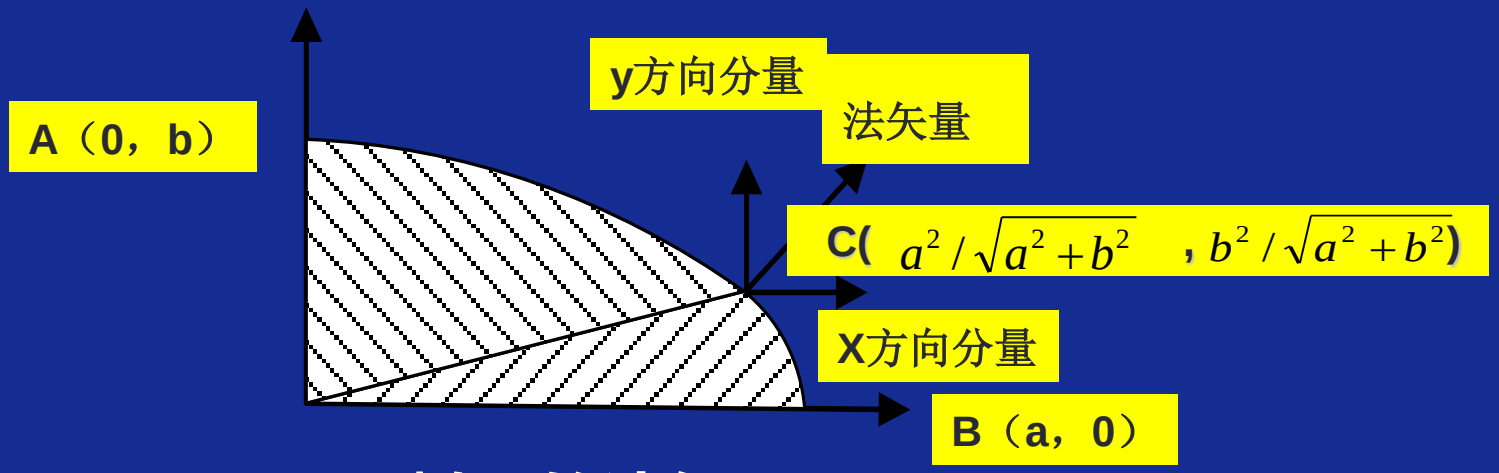
- 在处理第一象限的1/4椭圆弧时，进一步以法矢量两个分量相等的点把它分为两部分，上半部分Ⅰ和下半部分Ⅱ。该椭圆上一点P（x，y）处的法矢量为：

$$N(x,y)=\frac{\partial F}{\partial x}i+\frac{\partial F}{\partial y}j=2b^2xi+2a^2yj$$

式中，i和j是沿x轴向和沿y轴向的单位矢量。

3.3 椭圆的扫描转换

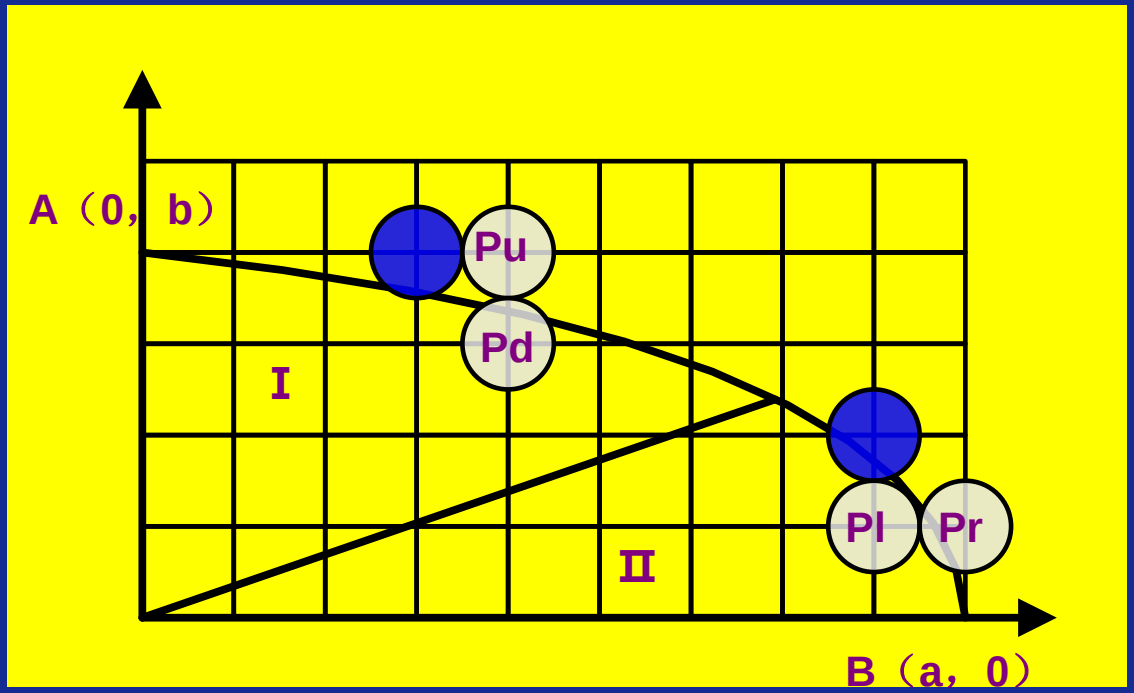
- 部分Ⅰ的AC椭圆弧段，法矢量的x向分量小于y向分量，斜率k处处满足|k|＜1，|△x|＞|△y|，所以x方向为主位移方向；
- 在C点，法矢量的x向分量等于y向分量，斜率k满足k＝－1，|△x|＝|△y|；
- 在部分Ⅱ的CB椭圆弧段，法矢量x向分量大于y向分量，斜率k处处满足|k|＞1，|△y|＞|△x|，所以y方向为主位移方向。



椭圆的法矢量

3.3 椭圆的扫描转换

- 椭圆的中点Bresenham算法的原理：
  - 在部分Ⅰ：每次在主位移x方向上走一步，y方向上退不退步取决于中点误差项的值；
  - 在部分Ⅱ：每次在主位移方向y上退一步，x方向上走不走步取决于中点误差项的值。



3.3 椭圆的扫描转换

- AC段椭圆弧。此时中Bresenham画椭圆算法要从A（0，b）到（ $a^2/\sqrt{a^2+b^2}$ ， $b^2/\sqrt{a^2+b^2}$ ）

顺时针确定顺时针确定最佳逼近于该段椭圆弧的像素点集。

- 由于x方向为主位移方向，假定当前点是P_i（x_i，y_i），下一步只能在正右方的像素P_u（x_i+1，y_i）和右下方的像素P_d（x_i+1，y_i-1）中选取。

3.3 椭圆的扫描转换

- CB段椭圆弧。此时中Bresenham画椭圆算法要从C（ $a^2/\sqrt{a^2+b^2}$ ， $b^2/\sqrt{a^2+b^2}$ ）到B（a，0）顺时针确定最佳逼近于该段椭圆弧像素点集。
- 由于y方向为主位移方向，假定当前点是P_i（x_i，y_i），下一步只能在正下方像素Pl（x_i，y_i-1）和右下方的像素Pr（x_i+1，y_i-1）中选取。

3.3.2 构造上半部分I 的中点误差项

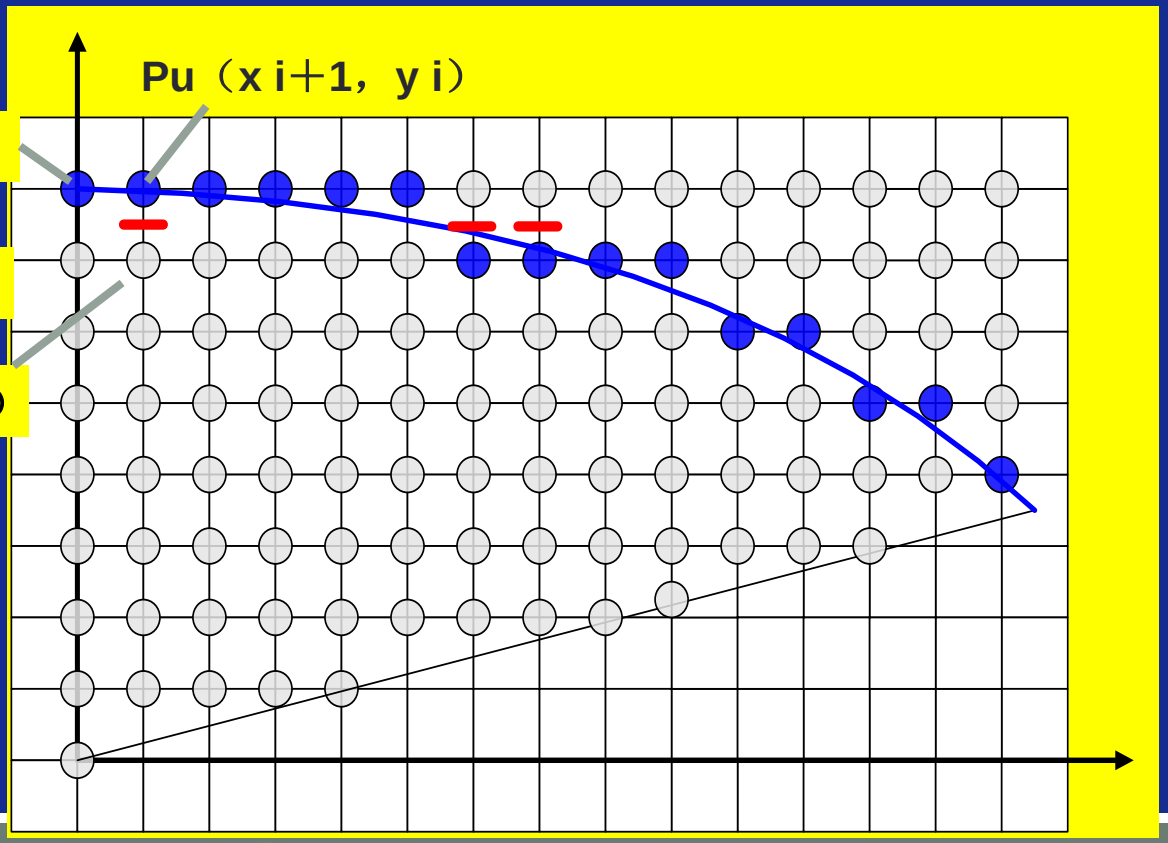
- 在上半部分 I ，x方向每次加1，y方向上减不减1取决于中点误差项的值。从P（x_i，y_i）走第一步，为了选取下一像素点的，需将P_u（x_i+1，y_i）和P_d（x_i+1，y_i-1）的中点M（x_i+1，$y_i-0.5$）代入隐函数，构造中点误差项：

$$d_1 = F(x_M, y_M) = F(x_i+1, y_i-0.5) = b^2(x_i+1)^2 + a^2(y_i-0.5)^2 - a^2b^2$$

  - 当$d_1<0$时，中点M在椭圆内，下一像素点应选择P_u，即y方向不退步；
  - 当$d_1>0$时，中点M在椭圆外，下一像素点应选择P_d，即y方向退一步；
  - 当$d_1 = 0$时，中点M在椭圆上，P_u、P_d和椭圆的距离相等，选择P_u或P_d均可，约定取P_d，如图3-13所示。

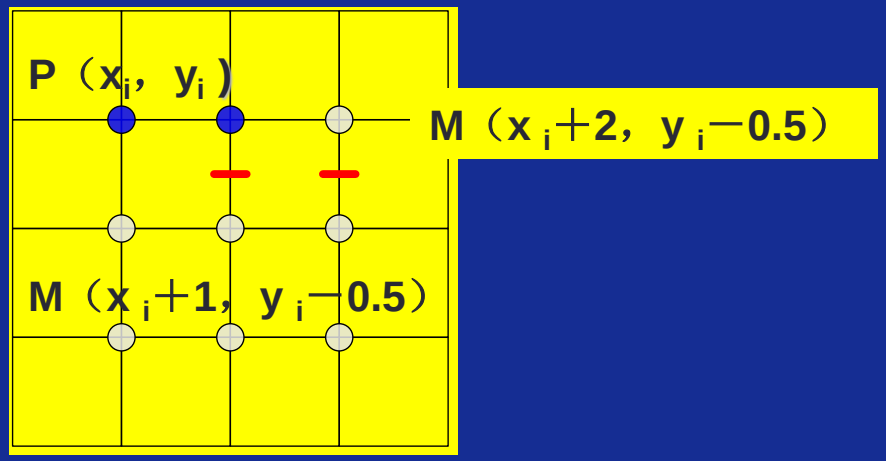
3.3.2 构造上半部分I 的中点误差项

$$y_{i+1} = \begin{cases} y_i & (d_1 < 0) \\ y_i - 1 & (d_1 \geq 0) \end{cases}$$

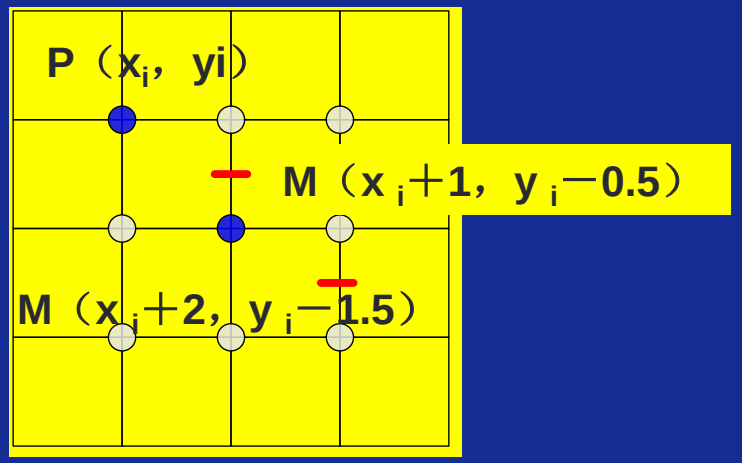


3.3.3 上半部分I的递推公式

- 为了能够继续判断椭圆上的每个点，需要给出中点误差项d_1的递推公式和初始值。



$d_1 < 0$



$d_1 \geq 0$

3.3.3 上半部分I的递推公式

- 现在如果考虑主位移方向再走一步，应该选取哪个中点代入中点误差项以决定应该选择的像素
  - (1)当$d_1<0$时，下一步的中点坐标为：
  - M（x_i+2，$y_i-0.5$）。所以下一步中点误差项为：

$$d_{1(i+1)}=F(x_i+2,y_i-0.5)=b^2(x_i+2)^2+a^2(y_i-0.5)^2-a^2b^2$$

$$=b^2(x_i+1)^2+a^2(y_i-0.5)^2-a^2b^2+b^2(2x_i+3)=d_{1i}+b^2(2x_i+3)$$

3.3.3 上半部分I的递推公式

- (2)当$d_1 \geq 0$时，下一步的中点坐标为：
- M（$x_i+2, y_i-1.5$）。所以下一步中点误差项为：

$$d_{1(i+1)} = F(x_i+2, y_i-1.5) = b^2(x_i+2)^2 + a^2(y_i-1.5)^2 - a^2b^2$$

$$= b^2(x_i+1)^2 + a^2(y_i-0.5)^2 - a^2b^2 + b^2(2x_i+3) + a^2(-2y_i+2)$$

$$= d_{1i} + b^2(2x_i+3) + a^2(-2y_i+2)$$

3.3.3 上半部分I的递推公式

- 中点误差项d_1的初值
  - 上半部分椭圆的起点为A（0，b），因此，第一个中点是（1，b - 0.5），对应的d_1的初值为：

$$d_{10} = F(1, b-0.5) = b^2 + a^2(b-0.5)^2 - a^2b^2$$
$$= b^2 + a^2(-b+0.25)$$

3.3.2 构造下半部分II 的中点误差项

- 在下半部分II，主位移方向发生变化。

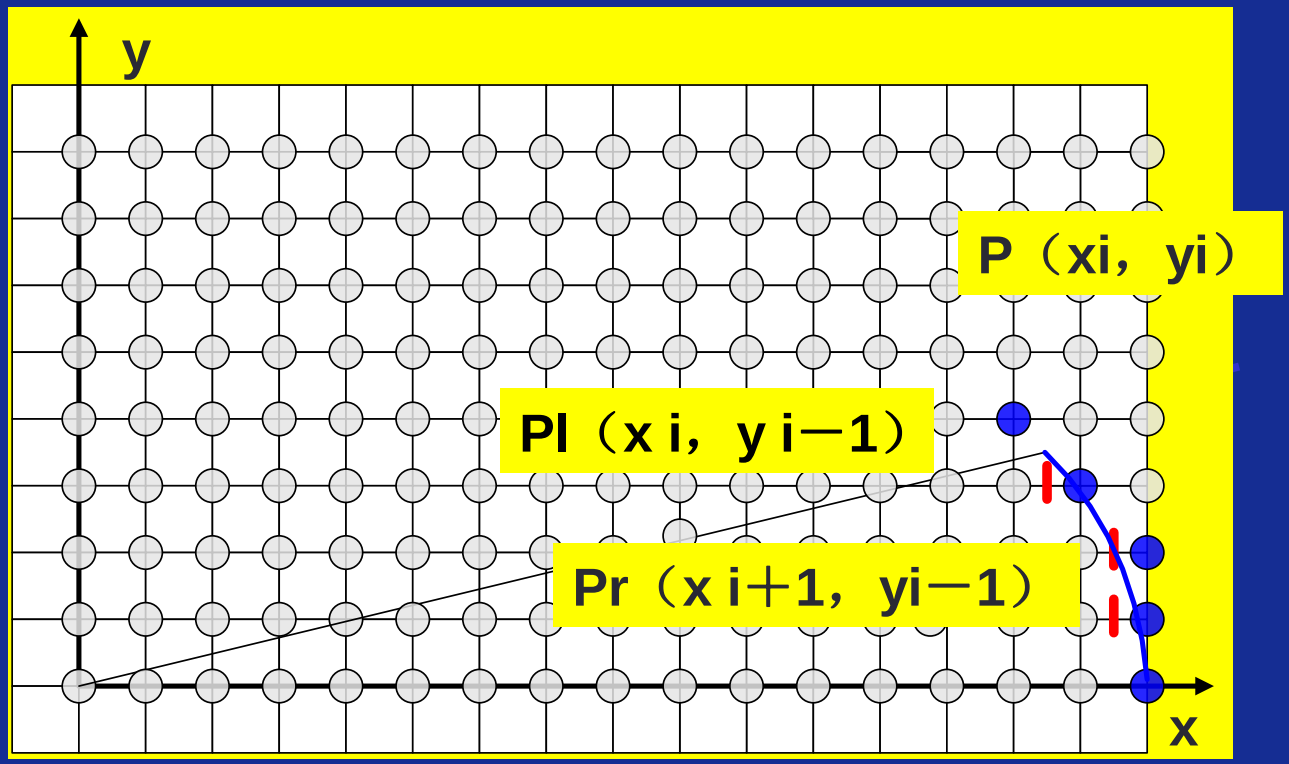
y方向每次减1，x方向上加不加1取决于中点误差项的值。从P_i（x_i，y_i）走第一步，为了选取下一像素点的，需将P_l（x_i，y_i - 1）和P_r（x_i + 1，y_i - 1）的中点M（x_i + 0.5，y_i - 1）代入隐函数，构造中点误差项：

$$d_2 = F(x_M, y_M) = F(x_i + 0.5, y_i - 1) = b^2(x_i + 0.5)^2 + a^2(y_i - 1)^2 - a^2b^2$$

- 当d_2<0时，中点M在椭圆内，下一像素点应选择Pr，即x方向上走一步；
- 当d_2>0时，中点M在椭圆外，下一像素点应选择P_l，即x方向上不走步；
- 当d_2 = 0时，中点M在椭圆上，P_l、P_r和椭圆的距离相等，选择P_l或P_r均可，约定取P_l，

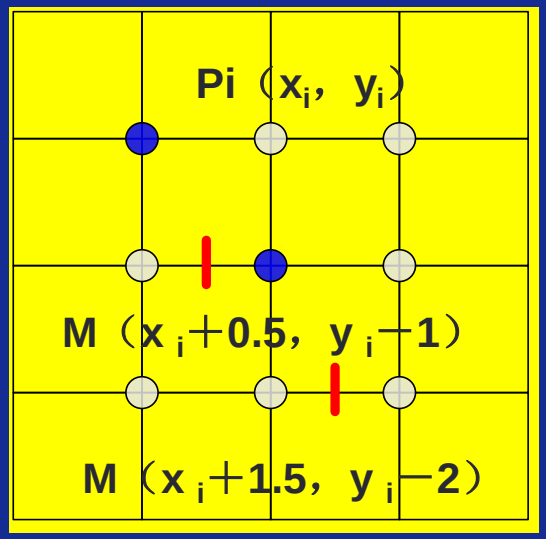
3.3.2 构造下半部分II 的中点误差项

$$x_{i+1} = \begin{cases} x_i + 1 & (d_2 < 0) \\ x_i & (d_2 \geq 0) \end{cases}$$

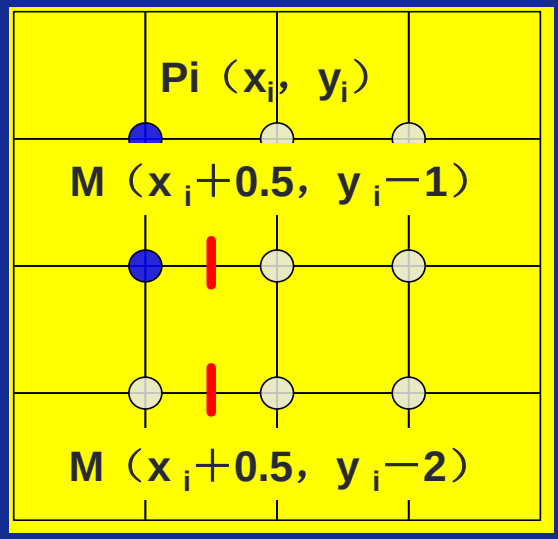


3.3.5 下半部分II的递推公式

- 为了能够继续判断椭圆上的每个点，需要给出中点误差项d_2的递推公式和初始值。



$d_2 < 0$



$d_2 \geq 0$

3.3.5 下半部分II的递推公式

- 现在如果考虑主位移方向再走一步，应该选取哪个中点代入中点误差项以决定应该选择的像素
  - (1)当$d_2<0$时，下一步的中点坐标为：
  - M（$x_i+1.5$，y_i-2）。所以下一步中点误差项为：

$$d_{2(i+1)}=F(x_i+1.5,y_i-2)=b^2(x_i+1.5)^2+a^2(y_i-2)^2-a^2b^2$$

$$=b^2(x_i+0.5)^2+a^2(y_i-1)^2-a^2b^2+b^2(2x_i+2)+a^2(-2y_i+3)$$

$$=d_{2i}+b^2(2x_i+2)+a^2(-2y_i+3)$$

3.3.5 下半部分II的递推公式

- (2)当$d_2 \geq 0$时，下一步的中点坐标为：
- M（x_i +0.5,y_i -2）。所以下一步中点误差项为：

$$d_{2(i+1)} = F(x_i + 0.5, y_i - 2) = b^2(x_i + 0.5)^2 + a^2(y_i - 2)^2 - a^2b^2$$

$$= b^2(x_i + 0.5)^2 + a^2(y_i - 1)^2 - a^2b^2 + a^2(-2y_i + 3)$$

$$= d_{2i} + a^2(-2y_i + 3)$$

3.3.5 中点误差项d_2的初值

- 中点误差项d_2的初值
  - 在上半部分Ⅰ，法矢量的x向分量小于y向分量；在C点，法矢量的x向分量等于y向分量；在下半部分Ⅱ，法矢量的x向分量大于y向分量：
  - x向分量为：$2b^2x$ y向分量为：$2a^2y$
- 则对于上半部分椭圆上一点任意P（x_i，y_i）如果在其当前中点M（x_i+1，$y_i-0.5$）处，满足x向分量小于y 向分量：

$$b^2(x_i+1)<a^2(y_i-0.5)$$

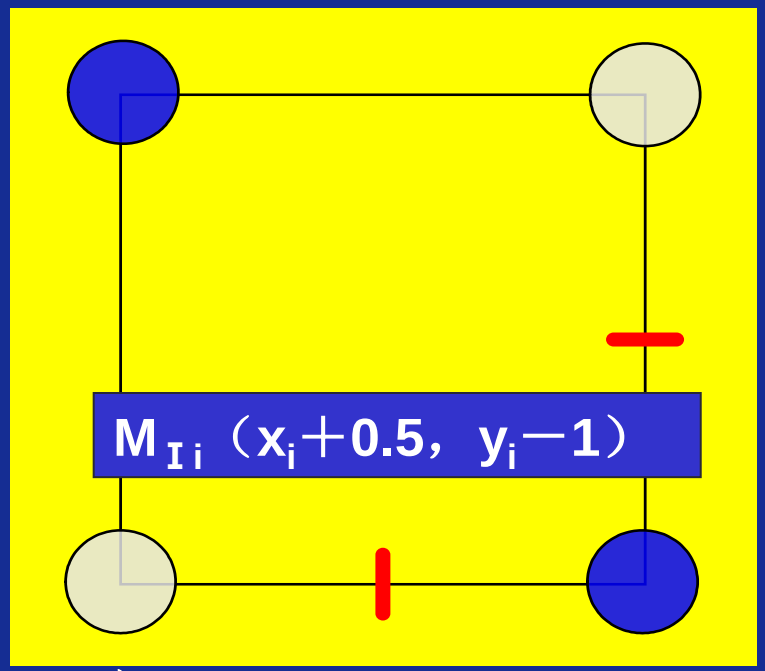
而在下一个中点，不等号改变方向，则说明椭圆从上半部分I转入了下半部分II

3.3.5 中点误差项d_2的初值

- 假定P（x_i，y_i）点是椭圆上半部分Ⅰ的最后一个像素，$M_Ⅰ$（x_i+1，$y_i-0.5$）是用于判断选择P_u和P_d像素的中点。
- 由于下一像素转入了下半部分Ⅱ，所以其中点改为判断P_l和P_r的中点$M_Ⅱ$（$x_i+0.5$，y_i-1），所以下半部分的初值d_{20}为：

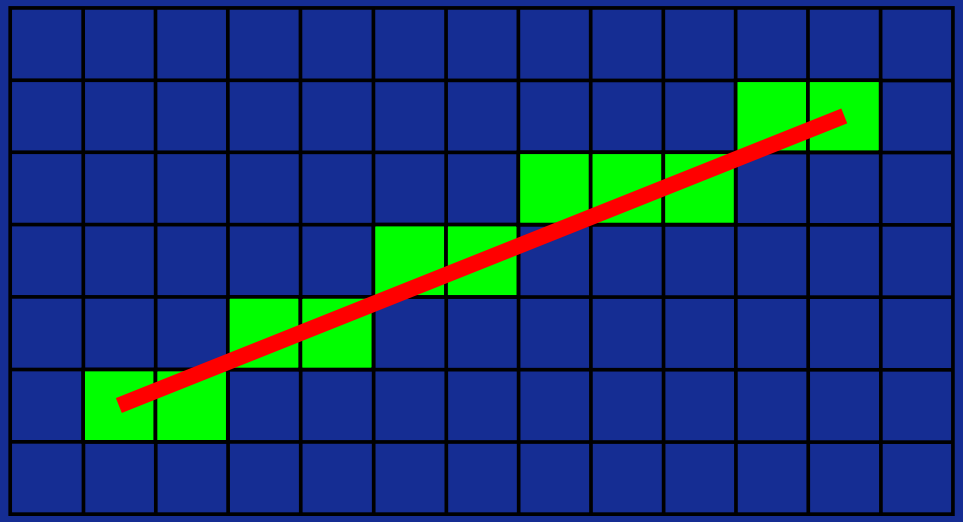
$$d_{20}=b^2(x+0.5)^2+a^2(y-1)^2-a^2b^2$$

3.3.5 中点误差项d_2的初值



反走样技术

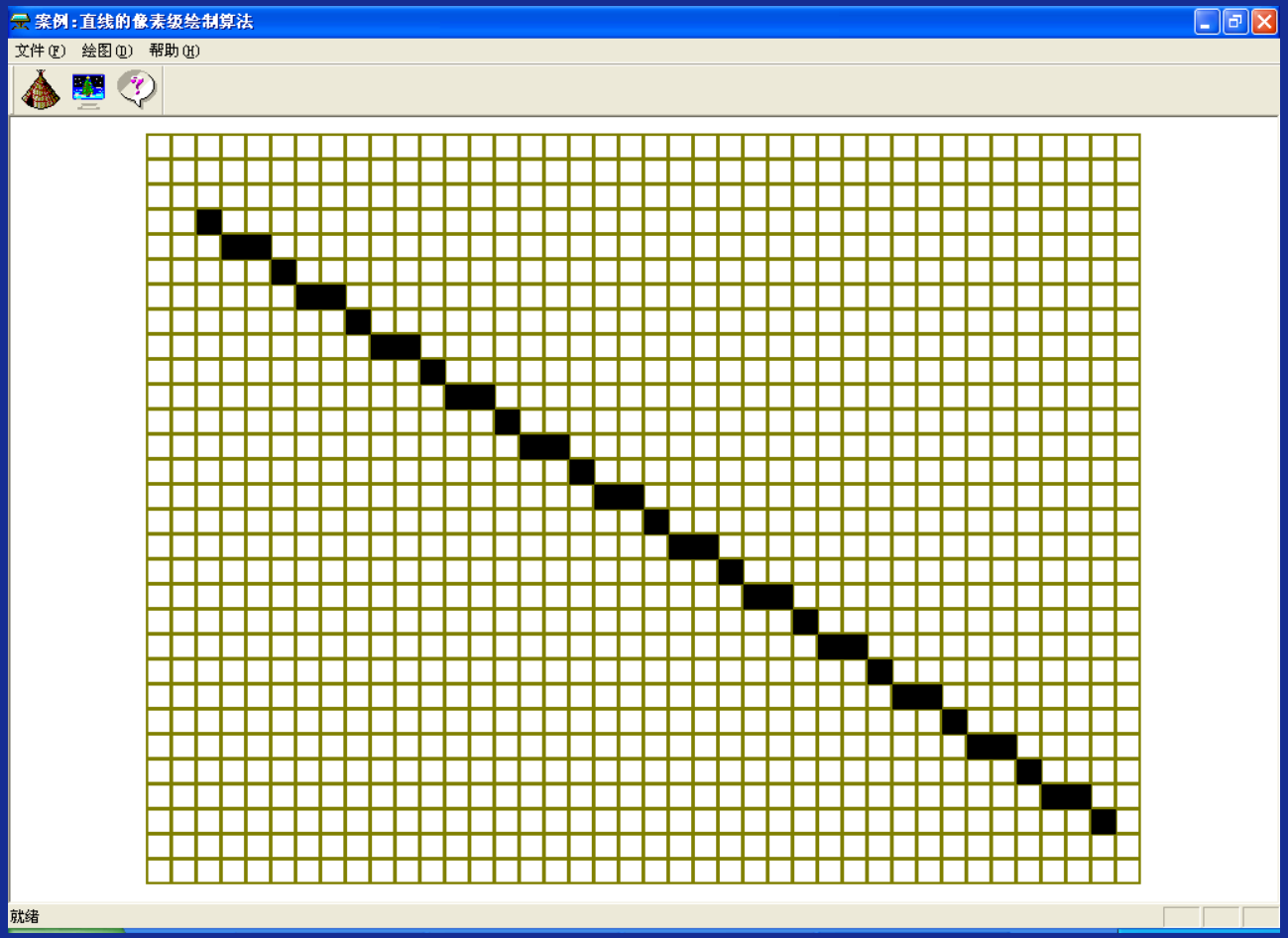
- 走样（Liasing）：用离散量表示连续量引起的失真
- 反走样（antialiasing）：用于减少或消除这种效果



绘制直线时的走样现象

直线扫描转换算法在处理**非水平非垂直的**直线时会出现**锯齿**,这是因为直线在光栅扫描器上显示的图像时一系列**亮度相同而面积不为零**的离散像素点。

反走样技术



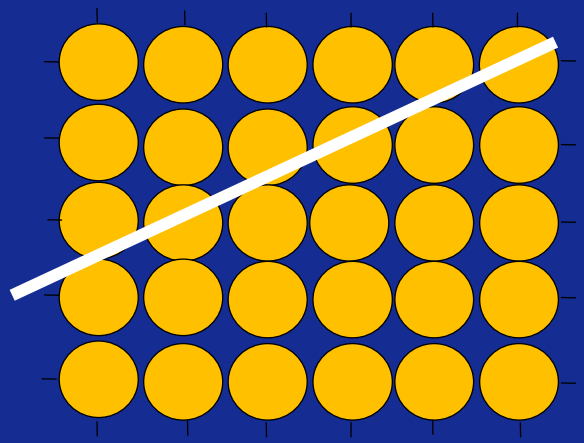
反走样技术

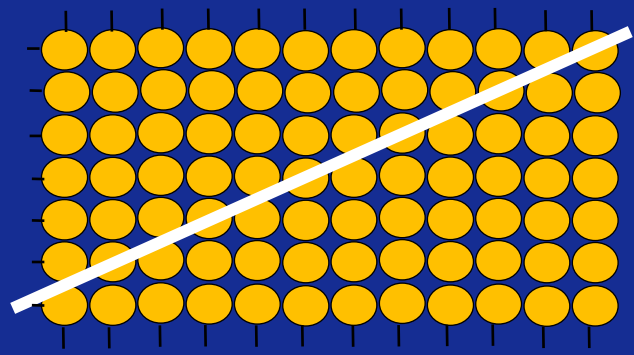
- 光栅图形的走样有如下几种:
  (1)产生阶梯或锯齿形;
  (2)细节或纹理绘制失真;
  (3)实时动画忽隐忽现、闪烁跳跃
- 反走样（antialiasing）：用于减少或消除这种效果
  1）提高分辨率方法
  2)Wu反走样技术
  3）非加权区域采样
  4）加权区域采样

反走样技术

提高分辨率

显示器的水平、竖直分辨率各提高一倍，则显示器的点距减少一倍，锯齿增加一倍，但同时每个阶梯的宽度也减小了一倍，所以显示出的直线段看起来就平直光滑了些。





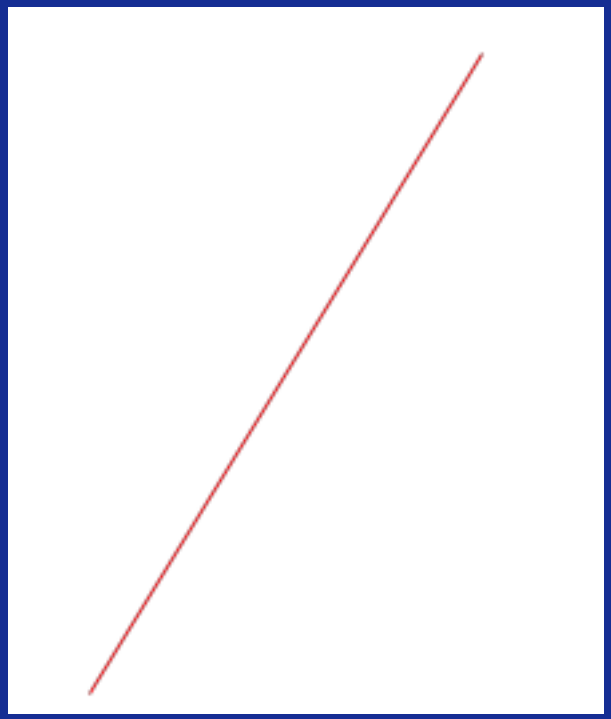
把显示器分辨率提高一倍后的结果

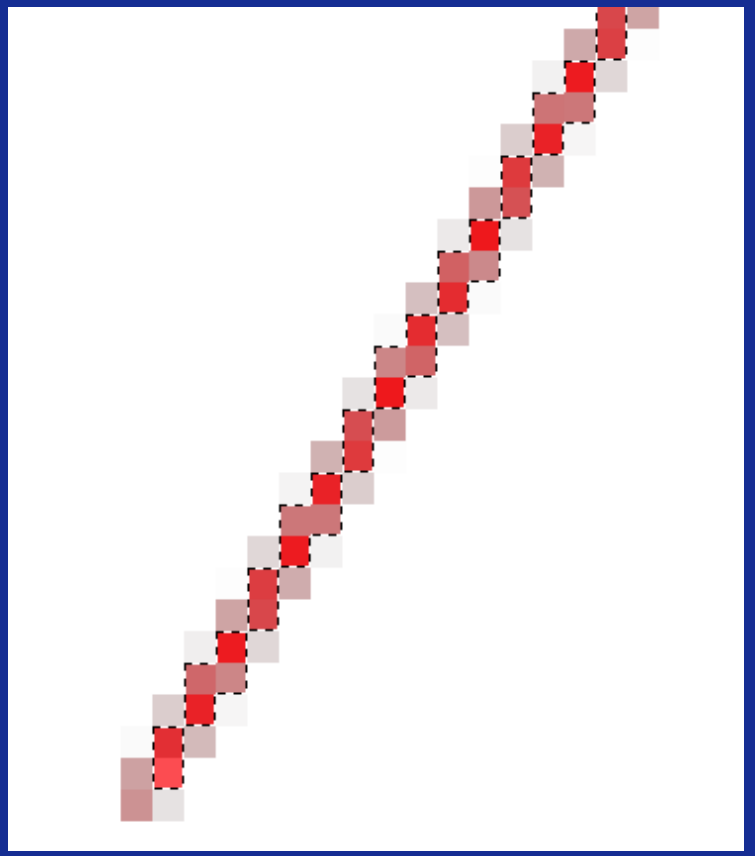
用中点算法扫描转换的一条直线

反走样技术

- 提高分辨率缺点三色荧光屏
- 这种反走样方法是以4倍的存储器代价和2倍的扫描转换时间获得的，并且只能减轻并不能消除锯齿问题。

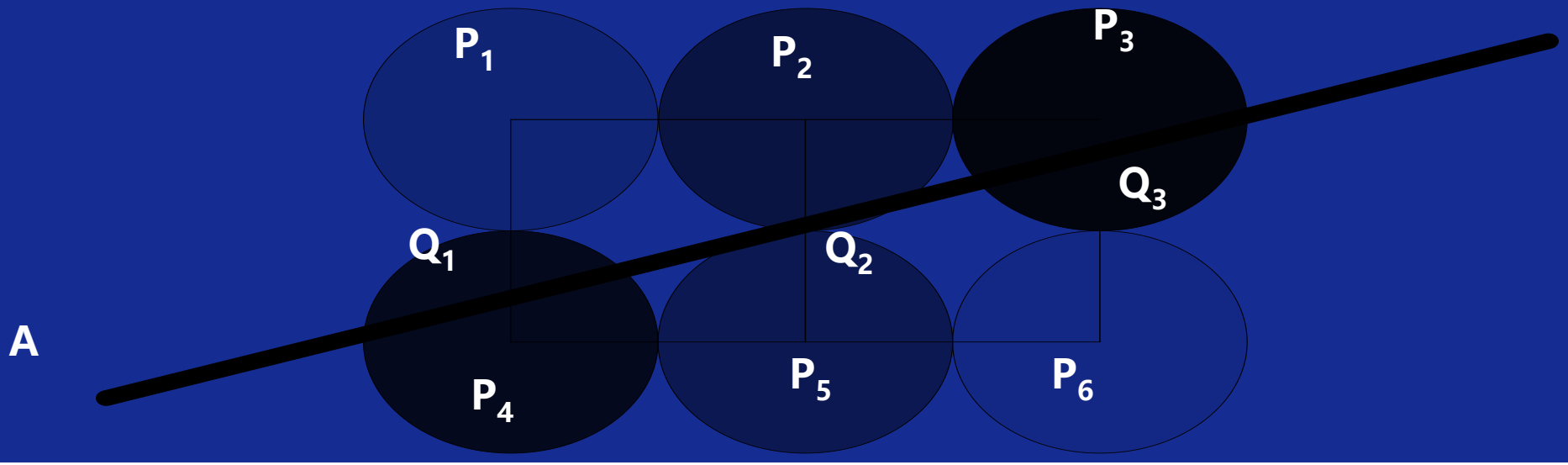
反走样技术





3.5直线反走样Wu算法

- 直线段AB经过扫描转换结果为P_4、P_2、P_3 其中P_4 显示在第一行，P_2、P_3 显示在第二行就出现了锯齿走样
- 原理：**对于理想直线的任意一点，同时用两个不同亮度等级的相邻像素来表示。**根据像素和理想直线的距离对相邻两个像素的亮度等级进行调节。
- 采用空间混色原理对走样进行修正



3.5直线反走样Wu算法

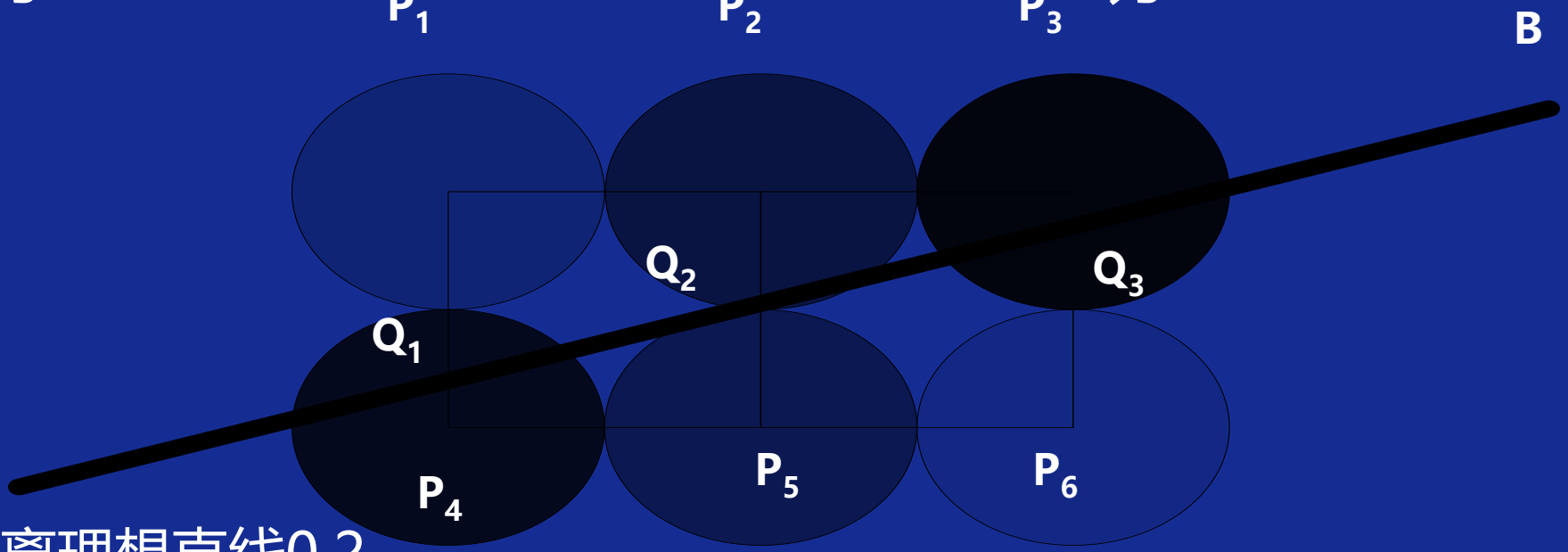
- 在RGB中（bRed、bGreen、bBlue）宏中，当 这bRed、bGreen、bBlue 3个值的变化率不同时，出现彩色；当bRed、bGreen、bBlue 3个值变化率相同时，出现不同等级的亮度。这3个值的分量的值都在0-255之间，共有256种亮度等级，并且规定亮度等级越大，像素越亮；亮度等级越小，像素越暗。
- 对于理想直线段上的Q1点扫描转换后可以由像素点P1和像素点P4已不同的亮度等级共同表示，像素点离理想直线段越近，亮度值越小，像素越暗；像素点离理想直线段越远，其亮度值越大，像素越亮，但二者的亮度之和等于直线段上的Q1的亮度值。

3.5直线反走样Wu算法

距离理想直线0.8个像素远的像素灰度为80%

距离理想直线0.45个像素远的像素灰度为45%

距离理想直线0.1个像素远的像素灰度为10%



距离理想直线0.2个像素远的像素灰度为20%

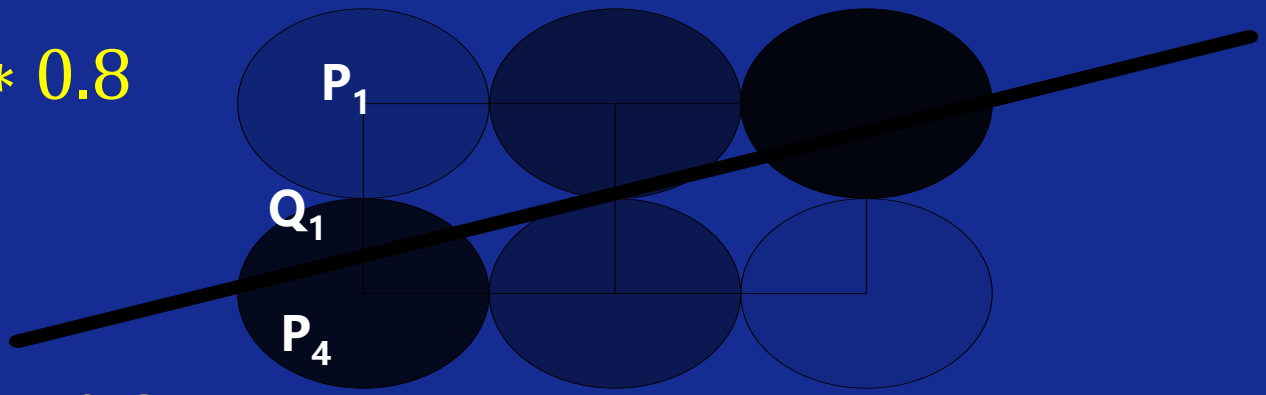
距离理想直线0.55个像素远的像素灰度为55%

距离理想直线0.9个像素远的像素灰度为90%

3.5直线反走样Wu算法

- 把白色与黑色之间按对数关系分成若干级，称为“灰度等级”。范围一般从0到255，白色为255，黑色为0，故黑白图片也称**灰度图像**。

$$I_{p1} = I * 0.8$$



$$I_{p4} = I * 0.2$$

$$I = I_{p1} + I_{p4} = I * 0.8 + I * 0.2$$

- 距离越远，权重越大，灰度越大，越靠近亮色
- 距离越近，权重越小，灰度越小，越靠近暗色

3.5直线反走样Wu算法

- 把白色与黑色之间按对数关系分成若干级，称为“灰度等级”。范围一般从0到255，白色为255，黑色为0，故黑白图片也称灰度图像，在医学、图像识别领域有很广泛的用途。考虑将偏差e作为加权参数，同时用上下（或左右）两个像素点的颜色来表示交点f的颜色。像素a的颜色为c1＝RGB（e×255，e×255，e×255），像素b的颜色为c2＝RGB（（1 - e）×255，（1 - e）×255，（1 - e）×255），两个像素点颜色的各个相应分量级别之和等于255。
- 当偏差e越小时，c1的分量e×255值越小，颜色越暗，同时c2的分量（1 - e）×255值越大，颜色越亮；反之亦然。用两个像素来表示理想线条上的一个点，并依据两个像素与理想直线的距离而调节其灰度级别，所绘制的线条可以达到视觉上消除阶梯的效果。

3.5直线反走样Wu算法

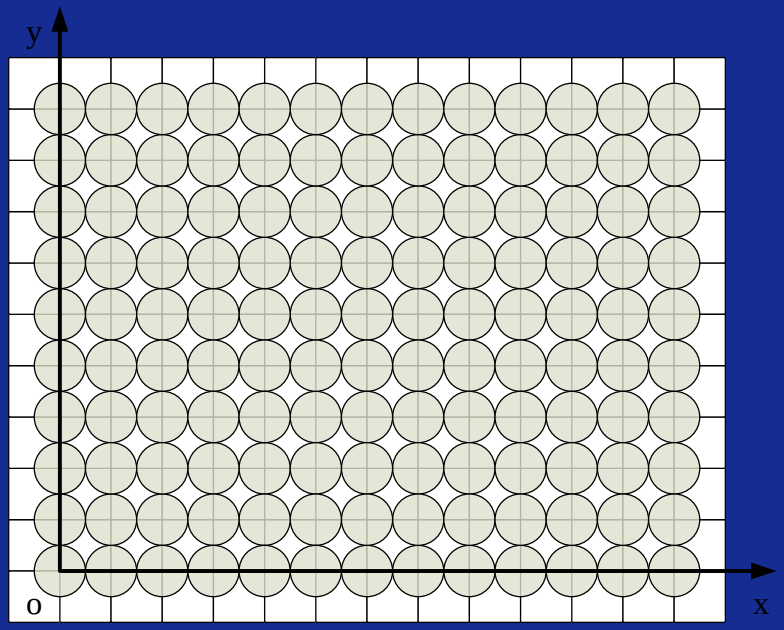
- 用两个像素来共同表示理想直线段上的一个点，依据两个像素与理想直线的距离而调节其亮度等级，使所绘制的直线达到视觉上消除锯齿的效果。
- 实际使用中，两个像素宽度的直线反走样的效果最好，视觉上效果上直线的宽度会有所减少，看起来就像一个像素宽度的直线。

本章小结

- 本章主要讲解了直线、圆和椭圆的中点Bresenham算法原理以及直线的反走样技术。
- 直线、圆和椭圆作为图形基本图元，其生成算法的优劣对整个图形系统的效率至关重要。在像素级生成算法中，中点Bresenham 算法避免了复杂运算，使用了最小的计算量，使单点基本图形生成算法已无优化的余地，已经成为使用最广泛的扫描转换算法。
- 走样是直线光栅化扫描转换的必然结果，只能减轻，不可避免，如何设计直线的反走样算法是计算机图形学的前沿研究课题。

3.7 习题

1.计算起点坐标为（0,0），终点坐标（12,9）直线的中点Bresenham算法的每一步坐标值以及中点误差项d的值（请写出中点误差项的递推公式和初始值），填入表3-2中，并用黑色点亮图3-29中的直线像素。



章节测评

1、什么是计算机图形学？（20分）

2、什么是光栅化？什么是扫描转换？（20分）

3、什么是反走样？（20分）

4、请写出中点Bresenham画线法的中点误差项递推公式（40分）

章节测评

1、什么是计算机图形学?

答：研究如何利用计算机表示、生成、处理和显示图形的学科。

2、什么是光栅化？什么是扫描转换?

答：基本图形光栅化就是在像素点阵中确定最佳逼近于理想图形的像素点集，并用制定颜色显示这些像素点集的过程。

当光栅化与按扫描线顺序绘制图形的过程结合在一起时，称为扫描转换。

3、什么是反走样?

答：用于减轻走样现象的技术。

4、请写出中点Bresenham画线法的中点误差项递推公式

答：见书P90，公式3-4和3-5.